Attribute-Based Multi-Input FE (and more) for Attribute-Weighted Sums

Shweta Agrawal[1], Junichi Tomida[2], and Anshu Yadav[1]

[1] IIT Madras, Chennai, India
shweta@cse.iitm.ac.in, anshu.yadav06@gmail.com
[2] NTT Social Informatics Laboratories
tomida.junichi@gmail.com

Abstract. Recently, Abdalla, Gong and Wee (Crypto 2020) provided the first functional encryption scheme for attribute-weighted sums (AWS), where encryption takes as input N (unbounded) attribute-value pairs $\{\mathbf{x}_i, \mathbf{z}_i\}_{i\in[N]}$ where $\mathbf{x}_i$ is public and $\mathbf{z}_i$ is private, the secret key is associated with an arithmetic branching programs f, and decryption returns the weighted sum $\sum_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i$, leaking no additional information about the $\mathbf{z}_i$'s.

We extend FE for AWS to the significantly more challenging multi-party setting and provide the first construction for *attribute-based* multi-input FE (MIFE) supporting AWS. For $i \in [n]$, encryptor i can choose an attribute $\mathbf{y}_i$ together with AWS input $\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}$ where $j \in [N_i]$ and N_i is unbounded, the key generator can choose an access control policy g_i along with its AWS function h_i for each $i \in [n]$, and the decryptor can compute

$$\sum_{i\in[n]} \sum_{j\in[N_i]} h_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j} \text{ iff } g_i(\mathbf{y}_i) = 0 \text{ for all } i \in [n]$$

Previously, the only known attribute based MIFE was for the inner product functionality (Abdalla *et al.* Asiacrypt 2020), where additionally, $\mathbf{y}_i$ had to be fixed during setup and must remain the same for all ciphertexts in a given slot.

Our attribute based MIFE implies the notion of multi-input *attribute based encryption* (MIABE) recently studied by Agrawal, Yadav and Yamada (Crypto 2022) and Francati, Friolo, Malavolta and Venturi (Eurocrypt 2023), for a conjunction of predicates represented as arithmetic branching programs (ABP).

Along the way, we also provide the first constructions of multi-client FE (MCFE)[3] and dynamic decentralized FE (DDFE) for the AWS functionality. Previously, the best known MCFE and DDFE schemes were for inner products (Chotard *et al.* ePrint 2018, Abdalla, Benhamouda and Gay, Asiacrypt 2019, and Chotard *et al.* Crypto 2020).

Our constructions are based on pairings and proven selectively secure under the matrix DDH assumption.

[3] The literature considers two notions termed as MCFE, one strictly stronger than the other. The stronger notion implies MIFE while the weaker does not. Here, we refer to the stronger notion, making MCFE a strict generalization of MIFE.

Table of Contents

1 Introduction

Multi-Party Functional Encryption. Functional encryption (FE) [SW05, BSW11] is a generalization of public key encryption which enables learning specific useful functions of encrypted data via "functional" keys. In FE, a secret key SK_f is associated with a function f, a ciphertext $\mathsf{CT}_{\mathbf{x}}$ is associated with a message $\mathbf{x}$ and decryption allows to compute $f(\mathbf{x})$ and nothing else. FE has been researched intensely in the community, with a long sequence of works that achieve increasingly powerful functionalities from diverse assumptions – please see [SW05, GPSW06, BSW07, KSW08, AFV11, ABV+12, GVW12, Wat12, GVW13, BGG+14, ABDP15, GVW15, Wee17a, AGW20, ACGU20, LL20a] and references therein.

While initially defined and constructed in the single input setting, i.e. with only one encryptor and one key generator, FE soon began to be generalized to distributed settings to capture the decentralized nature of both data and authority in the modern world. Computation on encrypted data generated independently at multiple sources, with fine-grained control on which data may be combined and with secret keys supporting decryption of meaningful aggregate functionalities, holds the promise of making FE much more relevant for real-world applications. These generalizations took different forms, from multi-input FE [GGG+14a] to multi-authority FE [Cha07] to multi-client FE [CDG+18a] to dynamic decentralized FE [CDSG+20] and such others [ACF+20]. These generalizations were captured via the abstraction of multi-party FE [AGT21], which sought to unify these different notions in a single framework.

The Attribute-Weighted Sums Functionality. Recently, Abdalla, Gong, and Wee [AGW20] introduced the functionality of *Attribute-Weighted Sums* (AWS) which supports computation of aggregate statistics on encrypted databases. Concretely, consider a database with N attribute-value pairs $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ where $\mathbf{x}_i$ is a public attribute associated with user i, and $\mathbf{z}_i$ is private. Given a function f, the AWS functionality on input $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ is defined as

$$\sum\nolimits_{i\in[N]} f(\mathbf{x}_i)^\top \mathbf{z}_i.$$

The AWS functionality is very natural, and Abdalla, Gong, and Wee suggested several compelling applications for it – for example, when f is a Boolean predicate then AWS can capture (i) the average salaries of minority groups holding a particular job title – here, $\mathbf{z}_i$ represents salary, while $f(\mathbf{x}_i)$ tests for membership in the minority group, (ii) approval ratings of an election candidate amongst specific demographic groups in a particular state – here, $\mathbf{z}_i$ is the rating, while $f(\mathbf{x}_i)$ computes membership in said group. Similarly, when $\mathbf{z}_i$ is Boolean, AWS can capture average age of smokers with lung cancer, where $\mathbf{z}_i$ is lung cancer and f computes average age.

Distributing the Data. In this work, we argue that for several applications of AWS, including the motivating examples provided by [AGW20], the data $(\mathbf{x}_i, \mathbf{z}_i)_{i\in[N]}$ is likely to be distributed across multiple sources which must compute ciphertexts independently. Concretely, in the example of computing average salaries of minority groups holding a particular job title, the data about the individuals would be generated across organizations, which are unlikely to even be in the same location. Similarly, when we compute whether a user is in a specific demographic group in a particular state, it is natural that user data would be distributed across different states, indeed even across different cities in a given state. In the third example, data about patients with lung cancer will naturally be generated and maintained at different hospitals that offer treatment for lung cancer, which would again be distributed geographically.

Thus, to capture data generation by independent sources, we extend FE for AWS to the multi-party setting. Concretely, we focus on the following primitives:

1. *Multi-Input FE (MIFE):* The primitive of multi-input FE (MIFE) [GGG+14b] allows the input to a function to be distributed among multiple (say n) parties. In more detail, the i^{th} party encrypts

its input $\mathbf{z}_i$ to obtain CT_i, a key authority holding a master secret generates a functional key SK_f and these enable the decryptor to compute $f(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ and nothing else.

We consider a further generalization of MIFE, namely *attribute-based* MIFE introduced by Abdalla *et al.* [ACGU20], which enables greater control on the leakage inherent by the functionality of MIFE, making it more suitable for practical applications. In an AB-MIFE for some functionality f, an attribute $\mathbf{y}_i$ is associated with a ciphertext for slot i, in addition to an input $\mathbf{z}_i$. The secret key is associated with an access control policy g in addition to the function input $\mathbf{c}$. Decryption first checks if $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = 1$, and if so, it computes the MIFE functionality $f(\{\mathbf{z}_i\}, \mathbf{c})$.

2. *Multi-Client FE (MCFE):* MCFE [GGG+14b, CDG+18a, CDG+18b] is a generalization of MIFE. In MCFE, the inputs $\mathbf{z}_i$ are additionally associated with public "labels" L_i and inputs can only be combined with other inputs that share the same label. As in MIFE, a functional key SK_f is provided which allows the decryptor to compute $f(\mathbf{z}_1, \ldots, \mathbf{z}_n)$ as long as the corresponding labels match, i.e. $L_1 = \ldots = L_n$.
3. *Dynamic Decentralized FE (DDFE):* DDFE [CDSG+20], as the name suggests, is a decentralized variant of FE, where not only can ciphertexts be generated locally and independently but so can the keys. In DDFE for some functionality f, the setup step is localized and run independently by users, letting them generate their private and public keys individually. During encryption, the set of users with whom a given input or key object should be combined can be chosen dynamically. In more detail, each party can specify the set of parties with which its input may be combined, a label that controls which values should be considered together and the input $\mathbf{z}_i$ itself. Similarly, every user can also generate a key object which specifies the set of parties with which the key may be combined, and a key vector $\mathbf{c}_i$. For decryption, the ciphertexts and keys from the parties who mutually agree to combine their inputs and keys are put together to compute $f(\{\mathbf{z}_i\}_i, \{\mathbf{c}_j\}_j)$.

   Note that DDFE implies MCFE which implies MIFE[4].

Prior Work. We summarize the state of the art below.

The AWS Functionality. For the AWS functionality, even the weakest multi-input notion, namely MIFE is not known to the best of our knowledge. We note Abdalla *et al.* [AGW20] did propose a multi-party extension to their FE for AWS. However, this scheme is a much weaker primitive than the standard notion of MCFE (or even MIFE), since this scheme natively only supports a single ciphertext query per slot. To extend it the setting of multiple queries, the authors make use of non-interactive MPC to enable the parties to obtain a random secret sharing of 0.

In more detail, while their scheme supports labels, the difference from standard MCFE schemes is that in their scheme each party uses a one-time secret key for each encryption instead of long-term encryption key, and the one-time keys are generated via non-interactive MPC run between the parties. Specifically, their scheme consists of five algorithms $(\mathsf{Setup}, \mathsf{OTSKGen}, \mathsf{Enc}, \mathsf{KeyGen}, \mathsf{Dec})$, and $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Dec}$ are the same as those in standard MCFE. $\mathsf{OTSKGen}(1^\lambda)$ is a non-interactive protocol where party i obtains one-time secret key otsk_i as the output of the protocol. $\mathsf{Enc}(\mathsf{otsk}_i, \mathbf{x}_i)$ takes otsk_i and a message $\mathbf{x}_i$ and outputs a ciphertext CT_i for party i. Correctness holds, i.e., decrypting a set $\{\mathsf{CT}_i\}_{i\in[n]}$ of ciphertexts with a secret key for f reveals $f(\mathbf{x}_1, \ldots, \mathbf{x}_n)$, only when the set of ciphertexts are generated under the one-time secret keys $\{\mathsf{otsk}_i\}_{i\in[n]}$ derived from a single running of $\mathsf{OTSKGen}(1^\lambda)$. The one-time secret-key can be used only once for encryption, otherwise security does not hold any more. Thus, this notion is even weaker than the variant of MCFE with one-time labeling restriction [CDG+18a] and in particular, does not imply MIFE.

Multi-Input Attribute Based Encryption. An attribute based MIFE scheme implies a multi-input attribute based encryption scheme as a special case. In an MIABE scheme, encryptor i encodes a secret message m_i together with an attribute $\mathbf{y}_i$. The function key encodes a circuit g so that decryption

[4] In this paper, we use the term MCFE as a generalization of MIFE, so that it allows multiple use of labels [CDG+18b]. In contrast, a weaker notion of MCFE where each label can be used only once does not imply MIFE [GGG+14b, CDG+18a].

outputs $(m_1, \ldots, m_n)$ if $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = 1$. The generalization to multi-input predicate encryption additionally allows to hide the attributes $\mathbf{y}_i$.

In this setting, Agrawal, Yadav and Yamada [AYY22] recently provided a construction for arbitrary predicates in NC_1 from pairings and Learning With Errors. Additionally, Francati *et al.* [FFMV23] also provided a multi-input predicate encryption scheme for a conjunction of predicates. Their construction supports the class P and is based on the Learning With Errors problem. Moreover, if the arity of the function is restricted to a constant, their security game also supports user corruptions. However, their construction does not support collusions, which is one of the most important and technically challenging aspects of designing attribute based encryption schemes.

AB-MIFE, MCFE and DDFE. For AB-MIFE, the best known attribute-based MIFE scheme is for the inner product functionality [ACGU20]. Moreover, in the AB-MIFE construction by Abdalla *et al.* [ACGU20], the ABE attribute $\mathbf{y}_i$[5] associated with the i^{th} slot is fixed in the setup phase and must remain the same for all ciphertexts, instead of being chosen dynamically by the encryptor for each encryption. For MCFE [ABG19] as well as DDFE [CDSG+20], the largest achievable function class is linear functions (or inner products), albeit with function hiding [AGT21].

1.1 Our Results

In this work, we significantly extend the reach of multi-input functional encryption schemes by providing the first AB-MIFE, MCFE and DDFE schemes that support the AWS functionality. Our constructions satisfy the standard (selective) indistinguishability based security and rely on the matrix DDH assumption on bilinear groups. We discuss each of these contributions below.

AB-MIFE for AWS: We provide the first attribute-based MIFE for the AWS functionality. In our AB-MIFE, each encryptor can choose an attribute $\mathbf{y}_i$ specific to its AWS input $\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$, the key generator can choose an access control policy g_i along with its AWS function h_i for $i \in [n]$ and decryption computes:

$$f((\mathbf{y}_1, \{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}), \ldots, (\mathbf{y}_n, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]}))$$
$$= \begin{cases} \sum_{i\in[n]} \sum_{j\in[N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle & (g_1(\mathbf{y}_1) = \cdots = g_n(\mathbf{y}_n) = 0) \\ \perp & (\text{otherwise}) \end{cases}$$

Here, $\mathbf{y}_i, \mathbf{x}_{i,j}$ are public while $\mathbf{z}_{i,j}$ is private, and g_i, h_i belong to arithmetic branching programs (ABP). We note that the number of slots N_i for $i \in [n]$ can be unbounded, and chosen by the encryptor dynamically.

Connection with Multi-Input Attribute-Based Encryption. We observe that this functionality also implies the notion of multi-input attribute-based encryption (MIABE) [AYY22] for a conjunction of predicates represented as ABP. Thus, our construction supports the functionality $g(\mathbf{y}_1, \ldots, \mathbf{y}_n) = \bigwedge(g_i(\mathbf{y}_i) = 0)$, where each g_i is an ABP.

In contrast, the MIABE construction of [AYY22] supports an arbitrary $g \in \mathsf{NC}_1$ but only outputs a fixed message whereas our construction supports the AWS functionality. Additionally, our construction also supports a stronger security model which allows user corruption. In more detail, the MIABE construction of [AYY22] requires all encryptors to share the same master secret key, which obviously cannot be provided to the adversary. In contrast, different encryptors in our construction have different encryption keys, and we allow the adversary to obtain some of these in the security game. By applying the MIABE to MIPE compiler of [AYY22], we obtain a multi-input predicate encryption scheme for constant arity, albeit *without* support for user corruptions, due to the design of the compiler.

[5] In their notation, the embedding of access control policy and attribute are swapped to the ciphertext-policy setting – thus, for them $\mathbf{y}_i$ is an access control policy.

Work	Arity	Corruption	Collusion	Function Class	Assumption
[FFMV23]	Poly	No	No	Conjunctions in P	LWE
[FFMV23]	Constant	Yes	No	Conjunctions in P	LWE
[AYY22]	2	No	Yes	NC_1	Koala and LWE
This (MIABE)	Poly	Yes	Yes	Conjunctions in NC_1	Matrix DDH
This (MIPE)	Constant	No	Yes	Conjunctions in NC_1	Matrix DDH and LWE

Table 1. Comparison with Prior Work in MIABE and MIPE. Note that Koala is a non-standard knowledge type assumption on pairings. For [FFMV23] and [AYY22], the results for MIABE and MIPE are identical, but we achieve different results for the two. We consider CPA-1 sided security for the comparison with [FFMV23].

We also compare with the recent work of Francati *et al.* [FFMV23]. As discussed above, their MIABE construction supports no collusions and user corruptions only for constant (not polynomial) arity. In contrast, our construction of AB-MIFE supports unbounded collusions, as well as user corruptions for *polynomial* arity. However, our construction, being based on pairings, only supports the function class NC_1 while they support P. Additionally our construction supports computation of the expressive AWS functionality while theirs just recovers a fixed message (i.e. our scheme is an FE not an ABE). In the setting of MIPE, our construction does not support corruption but does support collusions, while theirs achieves the opposite. Please see Table 1 for a detailed comparison.

Multi-Client FE for AWS. We construct the first MCFE for Attribute-Weighted Sums, which generalizes MIFE described above. In more detail, each encryptor can choose input $\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$ together with a label L_i, the key generator can choose ABPs $\{f_i\}_{i\in[n]}$ and decryption computes:

$$f(\{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}, \ldots, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]}) = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$$

as long as all the L_i are equal. This is the first MCFE that supports a functionality beyond inner products to the best of our knowledge. Moreover, the number of slots N_i allowed to each party i are unbounded, though the number of parties n is bounded – this feature was not achieved by prior MCFE schemes for inner products as far as we are aware.

This functionality is captured by separable functions defined by Ciampi *et al.* [CSW21]. A multi-input function f is separable if it can be described as $f(x_1, \ldots, x_n) = f_1(x_1) + \cdots + f_n(x_n)$ for some functions $f_1, \ldots, f_n$. They constructed MCFE for separable functions where $f_1, \ldots, f_n$ belong to poly-size circuits from function-hiding FE for *general circuits*. Their scheme is secure against a priori bounded number of ciphertext queries. Our results differ significantly from theirs because our schemes are secure against unbounded number of ciphertext queries and use only the much weaker MDDH assumption.

Dynamic Decentralized FE for AWS. Next, we extend our MCFE to the much more challenging setting of DDFE. For the setting of AWS, the i^{th} encryptor chooses a set of users $\mathcal{U}_{M,i}$ with whom its input may be combined, some label L_i to constrain which values should be considered together, aside from its AWS inputs $\{\mathbf{z}_{i,j}\}_{j\in[N_i]}$ which are private and $\{\mathbf{x}_{i,j}\}_{j\in[N_i]}$ which are public. For key generation, the i^{th} user also chooses a set of users $\mathcal{U}_{K,i}$ and a set of ABPs $\bar{f}_i = \{f_j\}_{j\in\mathcal{U}_{K,i}}$. If all the sets $\mathcal{U}_{M,i}$ and $\mathcal{U}_{K,i}$ match up (to some $\mathcal{U}'_K$) and if the labels in all n ciphertext slots are equal, then decryption computes the AWS functionality. Formally, for $k_i = (\bar{f}_i, \mathcal{U}_{K,i})$ and $m_i = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}, \mathcal{U}_{M,i}, L_i)$,

Work	(Pub, Pri) CT	Key	Functionality
MIFE [AGT22]	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \otimes \mathbf{z} \rangle$
AB-MIFE [ACGU20]	$(\bot, \mathbf{z}_i)$	$\{y_i, \mathbf{c}_i\}_{i \in S}$	$\bigwedge_{i \in S} f_i(y_i) \cdot \sum_{i \in S} \langle \mathbf{z}_i, \mathbf{c}_i \rangle$
AB-MIFE, §4	$((\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_j), \{\mathbf{z}_{i,j}\}_j)$	$\{g_i, h_i\}_{i \in [n]}$	$\bigwedge_i (g_i(\mathbf{y}_i) = 0) \cdot \sum_{i \in [n]} \sum_{j \in [N_i]} h_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$
MCFE [CDG+18b, ABG19]	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \rangle$
MCFE, §5	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i \in [n]}$	$\sum_{i \in [n]} \sum_{j \in [N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$
DDFE, [CDSG+20, AGT21]	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \rangle$
DDFE, §6	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i \in S}$	$\sum_{i \in S} \sum_{j \in [N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$

Table 2. Prior state of the art and our results. We do not consider function hiding or MCFE schemes with only one time labels. Above, we denote $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ or $\mathbf{z} = (\mathbf{z}_i)_{i \in S}$. S is some subset of authorized users for a given key. A function f_i is a monotone span programs fixed in setup. Functions f_i, g_i, h_i are arithmetic branching programs chosen in key generation.

the functionality computes:

$$f'(\{i, k_i\}_{i \in \mathcal{U}'_K}, \{i, m_i\}_{i \in \mathcal{U}'_M}) = \begin{cases} \sum_{i \in \mathcal{U}'_K} \sum_{j \in [N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle & \text{if the conditions below are satisfied} \\ \bot & \text{otherwise} \end{cases}$$

The conditions are:

1. $\mathcal{U}'_K = \mathcal{U}'_M$ and $\forall\ i \in \mathcal{U}'_K, \mathcal{U}_{K,i} = \mathcal{U}_{M,i} = \mathcal{U}'_K$.
2. $\forall_{i,i' \in \mathcal{U}'_K}$, $\bar{f}_i = \bar{f}_{i'}$ and $L_i = L_{i'}$.

We summarize prior work in Table 2. Please see Appendix A for a more detailed summary. We explain our functionalities in the framework of multi-party FE [AGT21] in Appendix B.

1.2 New Applications

Our attribute-based MIFE enables several new and exciting applications that were not possible before. Let us begin with the example for AWS suggested by [AGW20], of computing average age of smokers who have lung cancer. In this case, the access control layer on top of the MIFE can capture the willingness of a user to even participate in such a study involving their medical data. For example, perhaps a user is willing to participate in this computation if certain criteria are satisfied, for instance if the study is being performed by doctors with certain specializations. Moreover, these criteria can be different for different users. This is exactly the kind of access control that an ABE system is designed to enforce. The required criteria can be specified by each user using its attribute $\mathbf{y}_i$ while the key holder's input g_i must encode her privileges so that she learns the AWS output only if $g_i(\mathbf{y}_i)$ is satisfied for all $i \in [n]$.

In the context of MIABE, Agrawal, Yadav and Yamada [AYY22] provided the following motivating example: a doctor is treating Covid patients and desires to understand the relation between Covid and other medical conditions such as asthma or cancer, each of which are treated at different locations. The records of a given patient are encrypted independently and stored in a central repository, and the doctor can be given a key that filters stored (encrypted) records according to criteria such as condition = 'Covid' and condition = 'asthma' and age group ='60-80' and enables decryption of these. Note that our AB-MIFE can already support a conjunction of predicates and suffices to enable the functionality of the above example. Moreover, in addition to supporting decryption of messages as in

MIABE, our AB-MIFE will even allow computing some aggregates on the private data, something beyond the capability of MIABE.

For MCFE and DDFE, generalizing inner products to AWS is clearly meaningful – all applications of AWS in the single input setting are meaningful in the setting with multiple users, with the additional expressiveness offered by MCFE and DDFE. For instance in DDFE, the number of users who can participate in a computation are unbounded and moreover, users can join dynamically – this is useful in real world applications such as the examples involving patients in the lung cancer study or users in the minority group discussed earlier.

1.3 Technical Overview

Recap of AGW. Our starting point is the functional encryption scheme by Abdalla, Gong, and Wee [AGW20], henceforth AGW, which provides the first construction supporting the AWS functionality for ABP from standard assumptions on bilinear maps. In more detail, the encryptor[6] computes a ciphertext encoding $\{\mathbf{x}_j, \mathbf{z}_j\}_{j\in[N]}$ where N is unbounded, $\mathbf{x}_j$ are public and $\mathbf{z}_j$ are private, the key generator computes a secret key encoding an ABP f and decryption recovers $\sum_{j\in[N]}\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle$. At a high level, their construction proceeds in two steps: (i) construct a single slot scheme, i.e. $N = 1$, which supports computation of $\langle f(\mathbf{x}), \mathbf{z}\rangle$, (ii) extend this to support unbounded N by running N instances of the single slot scheme, and cleverly handling leakage and size blowup that occurs along the way. As discussed by AGW, step (i) can be achieved by adapting a framework by Wee [Wee17b], and the main conceptual and technical novelty lies in achieving step (ii), especially in supporting unbounded N.

We review step (ii) next, as the ideas herein form the basis of our multi-input constructions. As discussed above, the first idea to handle $N > 1$ is to simply run N instances of the single slot scheme but this evidently does not work, since it would allow the decryptor to learn partial sums $\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle$ which are not revealed by the ideal functionality. To address this leakage, the single slot scheme is extended to handle "randomization offsets", namely to add masking values $w_j r$ to the partial sums, where w_j are sampled randomly by the encryptor such that $\sum w_j = 0$, and r is sampled randomly by the key generator. These masking values hide intermediate partial sums $\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle$, but when the partial sums are added, we recover $\sum_{j\in[N]}\langle f(\mathbf{x}_j), \mathbf{z}_j\rangle$ as desired.

To make the secret key size independent of N, AGW construct a hybrid argument over the N slots, collecting "partial sums" along the way – the details of this technique are not relevant for our purposes. They achieve selective simulation based security from the standard k-linear assumption over bilinear groups.

They then extend this construction to a setting where the N slots can be owned by N independent parties – to enforce the constraint that $\sum_{j\in[N]} w_j = 0$, they make use of a non-interactive MPC protocol where the parties communicate to generate these shares prior to each encryption. While this construction provides a first feasibility result for FE supporting the AWS functionality in the multi-party setting, it falls short of achieving the standard notion of MCFE in many important ways:

1. The MPC step introduces an additional round of interaction prior to each encryption[7] – this violates the primary demand of non-interaction that is placed on FE.
2. The ciphertexts constructed in different "iterations", i.e. generated via different runs of the MPC cannot talk to each other, thus failing to satisfy the main functionality requirement of even an MIFE scheme, which explicitly requires supporting such combinations. In more detail, consider a two slot MIFE scheme, where the first slot ciphertexts encode $\mathbf{x}^j$ for $j \in [Q_1]$, the second slot ciphertexts encode $\mathbf{y}^i$ for $i \in [Q_2]$ and the secret key encodes some function f. Then MIFE explicitly requires that $f(\mathbf{x}^j, \mathbf{y}^i)$ should be computable for any pair j, i. Indeed, the standard notion of MCFE *generalizes* MIFE by additionally supporting labels in each ciphertext that dictate how ciphertexts may be combined. The multi-party scheme of AGW does not imply an MIFE.

[6] Here we discuss the single input construction of AGW, the multi-input construction is discussed later.

[7] This can done in a offline phase, but then places a bound on the number of ciphertexts which can be computed.

3. The security game of the multi-client AGW construction does *not* handle the multi-challenge setting, which is the main technical challenge in any MIFE or MCFE construction. Indeed, handling the multi-challenge setting would *disallow* the usage of simulation security due to an impossibility result by Boneh, Sahai and Waters [BSW11] – since the AGW constructions satisfy simulation security, any generalization to the multi-input setting must necessarily take a different route.

Thus, the question of even constructing an MIFE for AWS, let alone generalizations to AB-MIFE, MCFE and DDFE, is completely open. We provide an outline of the AGW construction in Figure 1.

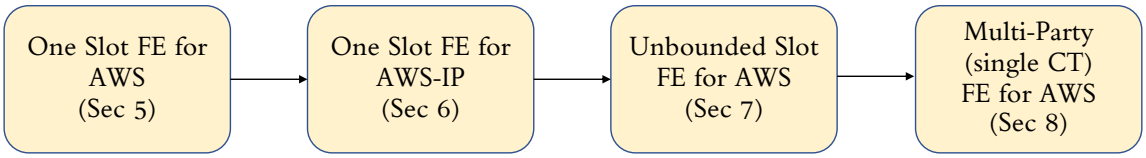

Fig 1. Construction Outline of AGW Multi-Client Scheme. All constructions satisfy simulation security. The multi-client scheme only supports a single ciphertext in each slot (and uses MPC to support many).

Building MIFE for AWS. In an MIFE for AWS, we have n parties, where the ciphertext computed by the i^{th} party embeds inputs $\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$ where N_i is unbounded, the secret key embeds a set of ABPs $\{f_i\}_{i\in[n]}$, and decryption computes

$$f(\{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}, \ldots, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]}) = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$$

Recall that $\mathbf{x}_i$ is public while $\mathbf{z}_i$ is private, i.e., a ciphertext only hides $\mathbf{z}_i$.

A natural idea would be to begin with the multi-party[8] construction of AGW and try to get rid of the MPC. In fact, removing the usage of MPC is not very difficult by using PRFs to compute a secret sharing of 0 for any given label[9], but this would still only lead to a weak variant of MCFE which has the so called "one time label" restriction. Intuitively, an MCFE with a one time label restriction, as the name suggests, allows each label to be used only one time for each input; this primitive therefore no longer implies MIFE. Handling combinations of multiple ciphertexts is the core functionality of MIFE and forms the basis for most applications, so the one time label restriction is quite a significant limitation. Indeed, in the inner product setting, early constructions of MCFE suffered from the one time label restriction [CDG+18a] and were upgraded to full-fledged MCFE by follow-up work using nontrivial ideas [CDG+18b, ABG19].

Another point to note is the handling of unbounded slots for each client. Concretely, let us say there are n clients, and the i^{th} one chooses N_i (unbounded) internal slots for their data. Now, the AGW multi-party construction can easily handle unbounded N by instantiating $\sum_{i\in[n]} N_i$ nominal clients and having each client internally handle N_i of these. This trick does not work out of the box anymore in the MIFE setting due to the requirement of supporting combinations of all ciphertexts across slots.

[8] Since the AGW construction does not satisfy the standard definition of MCFE in several important ways as discussed above, we refer to their construction as a multi-party construction, in the sense of [AGT21].

[9] Consider the 3 party case. Let us say that parties have PRF keys (k_1, k_2), (k_2, k_3), and (k_3, k_1) respectively. Then we can use the fact for every label L, $F(k_1, L) + F(k_2, L), -F(k_2, L) + F(k_3, L), -F(k_3, L) - F(k_1, L)$ are pseudorandom shares of 0 [KDK11, ABG19].

Our Approach. Taking a step back, a natural approach is to ask whether existing transformations of FE from the single to multi-input setting for the inner product functionality can help us overcome the challenges faced in designing this generalization for AWS. Towards this approach, we observe that all IP-MIFE (or IP-MCFE) schemes in the literature are constructed by (explicitly or implicitly) running an IPFE scheme in parallel for each input and handling leakage issues along the way. At a high level, these works can be classified into two categories based on which property of the underlying IPFE is required in the security proof: 1) ciphertext homomorphism, e.g., [AGRW17, CDG+18b, ACF+18, ABG19] or 2) function-hiding security, e.g., [DOT18, Tom19, AGT21].

While IPFE schemes have ciphertext homomorphism (a ciphertext is a group element, and adding ciphertexts of $\mathbf{x}_1$ and $\mathbf{x}_2$ results in a ciphertext of $\mathbf{x}_1 + \mathbf{x}_2$), this is not the case in FE for AWS due to public inputs for ABPs. Since ABPs are not linear functions, it is hopeless to equip an FE scheme for AWS with ciphertext homomorphism. It is worth noting that the reason that the AB-MIFE scheme in [ACGU20] can handle only a limited form of access control, i.e., only secret keys are associated with attributes, and access control is done between the attributes and the public fixed policy, comes from the fact that their scheme relies on the former approach and cannot associate ciphertexts with attributes or a policy as the case of the single input AB-FE schemes.

Fortunately, the latter approach is not ruled out, and indeed, we show that it can be made to work even for the AWS functionality. We observe that works in the latter category use function-hiding security of the underlying scheme to obtain function-hiding in the resultant IP-MIFE scheme. In this work, however, we will use function-hiding for a completely different purpose – to transform a singe-input scheme into a multi-input scheme *without* relying on ciphertext homomorphism of the underlying scheme. In particular, we will not achieve function-hiding security in our final MIFE for AWS scheme.

Recap: Construction of IP-MIFE from IPFE. Our starting point is therefore the FE to MIFE transformation for inner products by Datta, Okamoto and Tomida [DOT18] (henceforth DOT), which we describe next. Recall that IP-MIFE supports functions $f : (\mathbb{Z}_p^d)^n \to G_T$ specified by $(\mathbf{c}_1, \ldots, \mathbf{c}_n) \in (\mathbb{Z}_p^d)^n$ and defined as $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = [\sum_{i\in[n]} \langle \mathbf{x}_i, \mathbf{c}_i \rangle]_T$. While the DOT IP-MIFE scheme is a direct construction based on pairings, it can be viewed as a generic construction from a function-hiding FE scheme for inner product (IPFE) as described in the next paragraph. Recall that in an IPFE scheme, the ciphertext and secret key are associated with $\mathbf{x} \in \mathbb{Z}_p^d$ and $\mathbf{c} \in \mathbb{Z}_p^d$ respectively, and decryption reveals $[\langle \mathbf{x}, \mathbf{c} \rangle]_T$. The function-hiding property guarantees that the secret key hides $\mathbf{c}$ along with hiding $\mathbf{x}$ in the ciphertext.

Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme with the vector being $d + 1$. Then the IP-MIFE scheme is constructed as follows. Setup generates master secret keys $\mathsf{iMSK}_1, \ldots, \mathsf{iMSK}_n \leftarrow \mathsf{iSetup}(1^\lambda)$ and sets $\mathsf{EK}_i = \mathsf{iMSK}_i, \mathsf{MSK} = \{\mathsf{iMSK}_i\}_{i\in[n]}$. Encryption of $\mathbf{x}_i$ for slot i computes $\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_i, (\mathbf{x}_i, 1))$ and outputs $\mathsf{CT}_i = \mathsf{iCT}_i$. Key generation, given input $(\mathbf{c}_1, \ldots, \mathbf{c}_n)$, randomly chooses $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$ such that $\sum_{i\in[n]} r_i = 0$, computes $\mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, (\mathbf{c}_i, r_i))$ for $i \in [n]$, and outputs $\mathsf{SK} = \{\mathsf{iSK}_i\}_{i\in[n]}$. Decryption outputs $\sum_{i\in[n]} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_i) = [\sum \langle \mathbf{x}_i, \mathbf{c}_i \rangle]_T$, since $\sum r_i = 0$. Here, the random element r_i is used to hide partial decryption values $\langle \mathbf{x}_i, \mathbf{c}_i \rangle$.

Let us now turn our attention to the security proof. Since we need neither function hiding nor adaptive security for the multi-input scheme in our purpose, we can make the proof much simpler than that by DOT as follows. We will denote $\mathsf{iEnc}(\mathsf{iMSK}_i, \mathbf{v})$ and $\mathsf{iKeyGen}(\mathsf{iMSK}_i, \mathbf{v})$ by $\mathsf{iCT}_i[\mathbf{v}]$ and $\mathsf{iSK}_i[\mathbf{v}]$, respectively. Now, in the original game, the adversary is given $\mathsf{iCT}_i[(\mathbf{x}_i^{j,\beta}, 1)]$ for the j-th challenge message $(\mathbf{x}_i^{j,0}, \mathbf{x}_i^{j,1})$ and $\{\mathsf{iSK}_i[(\mathbf{c}_i^\ell, r_i^\ell)]\}_{i\in[n]}$ for the ℓ-th secret key of $(\mathbf{c}_1^\ell, \ldots, \mathbf{c}_n^\ell)$, where β is the challenge bit. Thus, the goal of the proof is to delete the information of β from the ciphertexts in an indistinguishable manner. In what follows, we omit index ℓ for conciseness since all secret keys can be handled in the same manner.

The security proof uses two hybrids. In the first hybrid, the j-th ciphertext for slot i is changed to $\mathsf{iCT}_i[(\mathbf{x}_i^{j,0}, 1)]$ while all secret keys are changed to $\{\mathsf{iSK}_i[(\mathbf{c}_i, r_i + \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i \rangle)]\}_{i\in[n]}$ for *all* i, j. The indistinguishability of the original game and the first hybrid follows from the security of the

function-hiding IPFE scheme and the following constraint:

$$\langle \mathbf{x}_i^{j,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{j,0}, \mathbf{c}_i \rangle = \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i \rangle \quad \text{for all } i, j \tag{1.1}$$

which follows from the fact that the adversary can inherently learn $\langle \mathbf{x}_i^{j,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle$ from challenge queries (originally observed in [AGRW17, page 4]). In the second hybrid, all secret keys are changed to $\{\mathsf{iSK}_i[(\mathbf{c}_i, r_i)]\}_{i\in[n]}$ for all i, which readily follows from the fact that the two distributions are equivalent:

$$\{(r_1, \ldots, r_n) : r'_1, \ldots, r'_n \leftarrow \mathbb{Z}_p \text{ s.t. } \sum r'_i = 0, \;\; r_i = r'_i + \langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i \rangle\}$$
$$\{(r_1, \ldots, r_n) : r_1, \ldots, r_n \leftarrow \mathbb{Z}_p \text{ s.t. } \sum r_i = 0\}$$

This is because we have that $\sum_{i\in[n]}(\langle \mathbf{x}_i^{1,\beta}, \mathbf{c}_i \rangle - \langle \mathbf{x}_i^{1,0}, \mathbf{c}_i \rangle) = 0$ due to the admissibility condition on the queries. At this point, the advantage of the adversary is 0 since its view contains no information about β.

Generalizing the FE to MIFE to support AWS. Next, we show how to generalize the FE to MIFE compiler of DOT to handle the AWS functionality. In this step, we make use of an insight developed by AGW to handle unbounded slots, namely, to leverage a (single input) FE scheme that supports *unbounded-slot* AWS together with randomization offsets. In more detail, a ciphertext is associated with $(\mathbf{v}, \mathbf{p}) \in \mathcal{X} \times \mathbb{Z}_p^m$, a secret key is associated with $(F, \mathbf{q}) \in \mathcal{F} \times \mathbb{Z}_p^m$, and decryption reveals $[F(\mathbf{v}) + \langle \mathbf{p}, \mathbf{q} \rangle]_T$. Here, we assume that $\mathcal{F}$ is a set of functions belong to unbounded-slot AWS and $\mathcal{X}$ is its input space, but observe that the argument below can be applied to any function classes. For security, we require that both $\mathbf{p}, \mathbf{q}$ are hidden. In what follows, we call this functionality AWS with inner product (AWSw/IP).

We emphasize that while this is also the functionality achieved by AGW [AGW20][Sec 6], the security achieved by these is quite different: our construction must satisfy partially function hiding *indistinguishability based* security, while theirs satisfies simulation based security without function hiding. Additionally, our construction will support the additional inner product functionality with respect to unbounded-slot AWS, while AGW support it for only single-slot AWS.

Suppose we have a partially function-hiding FE scheme for the AWSw/IP functionality, denoted as $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$. Then, we can construct MIFE that supports functions $F : (\mathcal{X})^n \to G_T$ specified by $(F_1, \ldots, F_n) \in \mathcal{F}^n$ and defined as $F(\mathbf{v}_1, \ldots, \mathbf{v}_n) = [\sum_{i\in[n]} F_i(\mathbf{v}_i)]_T$ from aFE by following the template of DOT, as described next. Looking ahead, F can be instantiated to capture either AWS or attribute based AWS to obtain MIFE for AWS or AB-MIFE for AWS respectively. For instance, for MIFE, we set $\mathbf{v}_i = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}$, $F = (f_1, \ldots, f_n)$ where f_i are ABPs, and $F(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle$.

Construction 1 (MIFE for AWS).

$\mathsf{Setup}(1^\lambda)$**:** It outputs $\mathsf{EK}_i = \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$ and $\mathsf{MSK} = \{\mathsf{aMSK}_i\}_i$.
$\mathsf{Enc}(\mathsf{EK}_i, \mathbf{v}_i)$**:** It outputs $\mathsf{CT}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{v}_i, 1))$.
$\mathsf{KeyGen}(\mathsf{MSK}, (F_1, \ldots, F_n))$**:** It outputs $\mathsf{SK} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where $r_1, \ldots, r_n \leftarrow \mathbb{Z}_p$ s.t. $\sum r_i = 0$ and $\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (F_i, r_i))$.
$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})$**:** It outputs $\sum_{i\in[n]} \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i) = [\sum F_i(\mathbf{v}_i)]_T$.

The security proof is essentially the same as in the case of IP-MIFE, discussed above. We use the following two hybrids: in the first hybrid, the j-th ciphertext for slot i is changed from $\mathsf{aCT}_i[(\mathbf{v}^{j,\beta}, 1)]$ to $\mathsf{aCT}_i[(\mathbf{v}^{j,0}, 1)]$ while all secret keys are changed from $\{\mathsf{aSK}_i[(F_i, r_i)]\}_{i\in[n]}$ to $\{\mathsf{aSK}_i[(F_i, r_i + F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0}))]\}_{i\in[n]}$. In this step, we leverage the important observation that a constraint similar to Eq. (1.1) holds in MIFE for the function class we consider, where the final output is the summation of the output of each slot. Specifically, we have $F_i(\mathbf{v}_i^{j,\beta}) - F_i(\mathbf{v}_i^{j,0}) = F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})$ for all i, j. Hence we can use

the function-hiding security of aFE to change the second element of the function in secret keys from r_i to $r_i + F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})$ in a indistinguishable manner. In the second hybrid, we bring back all secret keys to the form $\{\mathsf{aSK}_i[(F_i, r_i)]\}_{i\in[n]}$. This transition is possible as the case of IP-MIFE, that is, we use the fact that the following distributions are equivalent:

$$\{(r_1,\dots,r_n) : r'_1,\dots,r'_n \leftarrow \mathbb{Z}_p \ \text{ s.t. } \sum r'_i = 0, \ \ r_i = r'_i + F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})\}$$
$$\{(r_1,\dots,r_n) : r_1,\dots,r_n \leftarrow \mathbb{Z}_p \ \text{ s.t. } \sum r_i = 0\}$$

which follows from the query condition $\sum_{i\in[n]}(F_i(\mathbf{v}_i^{1,\beta}) - F_i(\mathbf{v}_i^{1,0})) = 0$. At this point, the advantage of the adversary is 0.

Partial Function Hiding FE for AWS with Inner Product. It remains to construct the single input, unbounded slot FE scheme for the AWSw/IP functionality which satisfies partial function hiding. As discussed, the AGW scheme achieves simulation-based security but not function hiding. Our idea of extending AGW to function-hiding to the multi-challenge setting is to design AGW using a function-hiding IPFE scheme, which is inspired by the constructions of ABE for ABP and FE for AWS from (slotted) function-hiding IPFE in [LL20a, DP21].

Recall that AGW first constructs a one-slot scheme that can handle randomization offsets, the construction of which basically follows the ABE scheme by [Wee17b], and then converts it to an unbounded-slot scheme in a modular manner. The spirit of our construction follows their blueprint, that is, we first construct a function-hiding one-slot scheme that can handle randomization offsets using a function-hiding IPFE scheme, and then convert it to a unbounded-slot scheme. However, we present the unbounded construction directly since later we will need to extend this to attribute based FE for AWSw/IP, and the modular construction does not apply to that setting. To see this, note that in an attribute-based FE for AWSw/IP, an attribute is associated with an unbounded-slot message and how to deconstruct the attribute for a one-slot message is unclear.

The key building block of [LL20a, DP21] is the arithmetic key garbling scheme (AKGS), which is specialized for constructing attribute-based encryption schemes. In our work, however, we use the (extended) partially-garbling scheme (PGS) for ABP [IW14, AGW20] together with function-hiding IPFE since PGS is more suitable for FE that computes ABPs and AWSs directly. Informally, it uses an algorithm $\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t})$ that takes an ABP $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_1}$, a public string $\mathbf{x} \in \mathbb{Z}_p^{n_0}$, private strings $\mathbf{z} \in \mathbb{Z}_p^{n_1}$ and $\delta \in \mathbb{Z}_p$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$, and outputs

$$\mathbf{L} = (\langle \mathbf{L}_1\mathbf{t}, \mathbf{x}'\rangle + \delta, \langle \mathbf{L}_2\mathbf{t}, \mathbf{x}'\rangle, \dots, , \langle \mathbf{L}_s\mathbf{t}, \mathbf{x}'\rangle, \mathbf{z}[1] + \langle \mathbf{L}_{s+1}\mathbf{t}, \mathbf{x}'\rangle, \dots, \mathbf{z}[n_1] + \langle \mathbf{L}_t\mathbf{t}, \mathbf{x}'\rangle)$$

where $\mathbf{x}' = (\mathbf{x}, 1)$, and $s, t \in \mathbb{N}, \mathbf{L}_i \in \mathbb{Z}_p^{(n_0+1)\times(t-1)}$ are deterministically computed from f. The algorithm pgb satisfies:

Corrrectness: we can efficiently compute a vector $\mathbf{b}_{f,\mathbf{x}} \in \mathbb{Z}_p^t$ from $f, \mathbf{x}$ such that $\langle \mathbf{L}, \mathbf{b}_{f,\mathbf{x}}\rangle = \langle f(\mathbf{x}), \mathbf{z}\rangle + \delta$;
Security: we can efficiently simulate the distribution of $\mathbf{L}$ over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$ from $(f, \mathbf{x}, \langle f(\mathbf{x}), \mathbf{z}\rangle + \delta)$.

Then, we can construct FE for AWSw/IP as follows. Let iFE be a function-hiding IPFE scheme as above.

Construction 2 (FE for AWSw/IP).

$\mathsf{Setup}(1^\lambda)$**:** It outputs $(\mathsf{PP}, \mathsf{MSK}) = (\mathsf{iPP}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$.
$\mathsf{Enc}(\mathsf{MSK}, (\{\mathbf{x}_j, \mathbf{z}_j\}_{j\in[N]}, \mathbf{p}))$**:** It chooses $u_1, \dots, u_N, w_1, \dots, w_N \leftarrow \mathbb{Z}_p$ s.t. $\sum_{j\in[N]} w_j = 0$. It defines

$$\mathbf{X}_j = \begin{cases} (u_j\mathbf{x}'_j, \mathbf{z}_j, w_j, \mathbf{p}, \ 0^\rho) & (j = 1) \\ (u_j\mathbf{x}'_j, \mathbf{z}_j, w_j, 0^m, 0^\rho) & (j > 1) \end{cases}$$

and computes $\mathsf{iCT}_j \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, \mathbf{X}_j)$ for $j \in [N]$. It outputs $\mathsf{CT} = (\{\mathbf{x}_j, \mathsf{iCT}_j\}_j)$. Note that the last ρ entries are used only for the security proof.

$\mathsf{KeyGen}(\mathsf{MSK}, (f, \mathbf{q}))$**:** It chooses $r \leftarrow \mathbb{Z}_p$, $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$ and computes $\mathbf{L}_1, \ldots, \mathbf{L}_t$ from f. It defines

$$\mathbf{Y}_j = \begin{cases} (\mathbf{L}_j\mathbf{t}, 0^{n_1}, \ \ r, \mathbf{q}, \ \ 0^\rho) & (j = 1) \\ (\mathbf{L}_j\mathbf{t}, 0^{n_1}, \ \ 0, 0^m, 0^\rho) & (1 < j \leq s) \\ (\mathbf{L}_j\mathbf{t}, \mathbf{e}_{j-s}, 0, 0^m, 0^\rho) & (s < j \leq t) \end{cases}. \tag{1.2}$$

where $\mathbf{e}_i$ is one-hot vector with the i-th element being 1. Finally it computes $\mathsf{iSK}_j \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, \mathbf{Y}_j)$ for $j \in [t]$ and outputs $\mathsf{SK} = (f, \{\mathsf{iSK}_j\}_j)$.

$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** It computes $[d_{j,\ell}]_T = \mathsf{iDec}(\mathsf{iCT}_j, \mathsf{iSK}_\ell)$ for all $j \in [N], \ell \in [t]$ and $\mathbf{b}_{f,\mathbf{x}}$ describe above. It outputs $\sum_{j\in[N]} \sum_{\ell\in[t]} [\mathbf{b}_{f,\mathbf{x}}[\ell] \cdot d_{j,\ell}]_T$.

In decryption, it follows that

$$(d_{j,1}, \ldots, d_{j,t}) = \begin{cases} \mathsf{pgb}(f, \mathbf{x}_j, \mathbf{z}_j, rw_j + \langle \mathbf{p}, \mathbf{q} \rangle; u_j\mathbf{t}) & (j = 1) \\ \mathsf{pgb}(f, \mathbf{x}_j, \mathbf{z}_j, rw_j; u_j\mathbf{t}) & (j > 1) \end{cases}$$

and thus $\sum_{j\in[N]} \sum_{\ell\in[t]} \mathbf{b}_{f,\mathbf{x}}[\ell] \cdot d_{j,\ell} = \sum_{j\in[N]} (\langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + rw_j + (j = 1)\langle \mathbf{p}, \mathbf{q} \rangle) = \sum_{j\in[N]} \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle$. Roughly speaking, the partially function-hiding security of this scheme follows from the following observations:

- Thanks to the function-hiding property of iFE, what the adversary can learn from CT and SK is $\{[(d_{j,1}, \ldots, d_{j,t})]_T\}_{j\in[N]}$.
- The random tape $\{[u_j\mathbf{t}]_T\}_{j\in[N]}$ used to compute $\{d_{j,\ell}\}$ looks random under the SXDH assumption, and $(d_{j,1}, \ldots, d_{j,t})$ for each j appear to be generated by a fresh random tape.
- Thanks to the security of the PGS, the only information about $(\{\mathbf{z}_j\}, \mathbf{p}, \mathbf{q})$ contained in $(d_{j,1}, \ldots, d_{j,t})$ is $\langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + rw_j + (j = 1)\langle \mathbf{p}, \mathbf{q} \rangle$.
- Under the SXDH assumption, $\{[rw_j]_T\}_{j\in[N]}$ looks random with the constraint that the summation of these is $[0]_T$. Thus, the only information about $(\{\mathbf{z}_j\}, \mathbf{p}, \mathbf{q})$ in $\{[(d_{j,1}, \ldots, d_{j,t})]_T\}_{j\in[N]}$ is $\sum \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle$.

We remark that we can easily modify the scheme such that Enc and KeyGen take vectors $\mathbf{p}$ and $\mathbf{q}$ as a vector of group elements. We will use this property later in the overview for DDFE for AWS.

Attribute-Based MIFE for AWS. Next, we explain how to make the above MIFE for AWS construction attribute-based. At a high level, we do the following: (i) Make FE for AWSw/IP Attribute-Based, (ii) Use the FE to MIFE compiler discussed above to "lift" this to AB-MIFE for AWS. We expand on these below.

Step 1: Make FE for AWSw/IP Attribute-Based. We extend FE for AWSw/IP such that it incorporates an ABP predicate which controls decryption, similarly to attribute-based encryption. Specifically, we add a public vector $\mathbf{y}$ to the message in the ciphertext and a public ABP g to the function in the secret key, and allow decryption only when $g(\mathbf{y}) = 0$.

A naive idea is to define $\mathbf{x}'_j = (\mathbf{y}, \mathbf{x}_j), \mathbf{z}'_j = (v, \mathbf{z}_j)$ and $f'(\mathbf{x}') = (a \cdot g(\mathbf{y}), f(\mathbf{x}))$ where $a, v \leftarrow \mathbb{Z}_p$ and use $\{\mathbf{x}'_j, \mathbf{z}'_j\}$ and f' as inputs for encryption and key generation of FE for AWSw/IP, respectively. Note that f' is an ABP if f, g in turn are ABPs. Then, decryption outputs $[\sum_{j\in[N]} (av \cdot g(\mathbf{y}) + \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle) + \langle \mathbf{p}, \mathbf{q} \rangle]_T$. Since $[av \cdot g(\mathbf{y})]_T$ looks random if $g(\mathbf{y}) \neq 0$ under the SXDH assumption, the decryptor can learn $[\sum_{j\in[N]} \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T$ only when $g(\mathbf{y}) = 0$.

However, this idea does not work since a needs to be provided in the clear in the secret key for decryption, and this disallows the reliance on the SXDH assumption (recall that we need f and $\mathbf{x}$ to compute $\mathbf{b}_{f,\mathbf{x}}$ in the decryption of the FE for AWSw/IP). To avoid this, we directly embed a in $\mathbf{Y}_{s+1}$ so that we can perform decryption without the knowledge of a. Concretely, we define f' as $f'(\mathbf{x}') = (g(\mathbf{y}), f(\mathbf{x}))$ instead of $f'(\mathbf{x}') = (a \cdot g(\mathbf{y}), f(\mathbf{x}))$ and define $\mathbf{Y}_{s+1} = (\mathbf{L}_{s+1}, \underline{a}\mathbf{e}_1, 0, 0^m, 0^\rho)$ in

Eq. (1.2). Then, the decryption result is the same as the naive construction, which follows from the correctness of the PGS, but f' does not contain information about a in this construction.

The proof of function-hiding security of this scheme is inspired from the proof in AGW [AGW20, Section 7], but with the following key differences: 1) we need to prove IND-based function-hiding security in the secret-key multi-challenge setting while AGW proves SIM-based security in the public-key setting (thus not function-hiding); 2) Our scheme is attribute-based while AGW is not. Hence we need to handle secret key queries that cannot decrypt some challenge ciphertexts, and for such ciphertexts the function values in $\beta = 0$ and $\beta = 1$ can be different (β is the challenge bit). Please see Section 3 for details.

Step 2: AB-MIFE for AWS. Suppose we have an FE scheme aFE where a ciphertext is associated with $(c, \mathbf{v}, \mathbf{p})$ while a secret key is associated with $(k, F, \mathbf{q})$, and decryption reveals $[F(\mathbf{v}) + \langle \mathbf{p}, \mathbf{q} \rangle]_T$ if $\mathsf{P}(c, k) = 1$ for some predicate P and $\perp$ otherwise. We also assume that aFE is partially function-hiding, so that the ciphertext hides (a part of) $\mathbf{v}$ and $\mathbf{p}$, and the secret key hides $\mathbf{q}$.

At first glance, it seems that we can construct AB-MIFE for $\mathcal{F}'$ from aFE by using Construction 1, where $\mathcal{F}'$ consists of functions $F' : (\mathcal{C} \times \mathcal{X})^n \to G_T$ specified by $((k_1, F_1), \ldots, (k_n, F_n)) \in (\mathcal{K} \times \mathcal{F})^n$ and defined as

$$F'((c_1, \mathbf{v}_1), \ldots, (c_n, \mathbf{v}_n)) = \begin{cases} [\sum_{i \in [n]} F_i(\mathbf{v}_i)]_T & \mathsf{P}(c_i, k_i) = 1 \ \text{ for all } i \\ \perp & \text{otherwise} \end{cases}$$

Note that AB-MIFE for AWS corresponds to the case where $c = \mathbf{y}, \mathbf{v} = \{\mathbf{x}_j, \mathbf{z}_j\}_{j \in [N]}$, $k = g$ where g is an ABP, $\mathsf{P}(c, k) = 1$ iff $g(\mathbf{y}) = 0$, and F is specified by an ABP f and defined as $F(\mathbf{v}) = \sum \langle f(\mathbf{x}_j), \mathbf{z}_j \rangle$. We can also observe that aFE for the above setting corresponds to attribute-based FE for AWSw/IP.

However the above construction is insufficient due to the following reason. Let us consider a two-input scheme where an adversary obtains ciphertexts of $(c_1^1, \mathbf{v}_1^1)$ and $(c_1^2, \mathbf{v}_1^2)$ for slot 1 (denoted by $\mathsf{CT}_1^1, \mathsf{CT}_1^2$), a ciphertext of $(c_2^1, \mathbf{v}_2^1)$ for slot 2 (denoted by CT_2^1), and a secret key for $((k_1, F_1), (k_2, F_2))$ (denoted by SK) such that $\mathsf{P}(c_1^j, k_1) = 1$ for both $j \in \{1, 2\}$ while $\mathsf{P}(c_2^1, k_2) = 0$. Note that CT_i^j denotes the j-th ciphertext for slot i. In this case, the adversary should not obtain any information about private inputs, since the predicate of slot 2 is never satisfied. However, the adversary can learn $[F_1(\mathbf{v}_1^2) - F_1(\mathbf{v}_1^1)]_T$ in this construction (recall that it can learn $[F_1(\mathbf{v}_1^1) + r_1]_T$ and $[F_1(\mathbf{v}_1^2) + r_1]_T$ by decryption of aFE in slot 1). This is leakage which we need to avoid.

An important fact is that this leakage is inherent if the adversary additionally obtains a ciphertext of $(c_2^2, \mathbf{v}_2^2)$ for slot 2 (denoted by CT_2^2) such that $\mathsf{P}(c_2^2, k_2) = 1$. This is because it can learn $[F_1(\mathbf{v}_1^2) - F_1(\mathbf{v}_1^1)]_T$ by $\mathsf{Dec}(\mathsf{CT}_1^2, \mathsf{CT}_2^2, \mathsf{SK}) - \mathsf{Dec}(\mathsf{CT}_1^1, \mathsf{CT}_2^2, \mathsf{SK})$. By generalizing this observation, it turns out that such leakage appears only when the adversary obtains an illegitimate secret key, which cannot decrypt any combinations of ciphertexts that the adversary has. More formally, we say a secret key for $((k_1, F_1), \ldots, (k_n, F_n))$ is illegitimate if there exists slot i and the adversary does not have a ciphertext of $(c_i, *)$ for slot i such that $\mathsf{P}(c_i, k_i) = 1$. In other words, the above construction is secure in the model where the adversary never asks for illegitimate secret keys – we refer to this notion as security against legitimate keys.

Achieving Security against Any Keys. We next show how to remove this restriction and achieve security against any keys starting with a scheme secure against legitimate keys. Our idea is to encrypt all secret keys and allow the adversary to decrypt only legitimate secret keys. We achieve such a construction by leveraging an n-out-of-n secret sharing scheme and an attribute-based encryption scheme ABE for the *dual predicate* of P, denoted by $\overline{\mathsf{P}}$. Note that $\overline{\mathsf{P}} : \mathcal{K} \times \mathcal{C} \to \{0, 1\}$ is defined as $\overline{\mathsf{P}}(k, c) = 1 \Leftrightarrow \mathsf{P}(c, k) = 1$. We describe this conversion next.

Let $\mathsf{wmFE} = (\mathsf{wmSetup}, \mathsf{wmEnc}, \mathsf{wmKeyGen}, \mathsf{wmDec})$ be an AB-MIFE scheme for $\mathcal{F}'$ secure against legitimate keys. The setup algorithm generates n master secret keys $\mathsf{abMSK}_1, \ldots, \mathsf{abMSK}_n$ of ABE and sets $(\mathsf{abMSK}_i, \mathsf{wmEK}_i)$ as an encryption key for slot i. Encryption of $(c_i, \mathbf{v}_i)$ for slot i is the same as wmEnc except that it appends a secret key of ABE for c_i to wmCT_i. Key generation of $\{k_i, F_i\}$ runs

$\mathsf{wmSK} \leftarrow \mathsf{wmKeyGen}(\mathsf{wmMSK}, \{k_i, F_i\})$, secret shares wmSK to $\sigma_1, \ldots, \sigma_n$, encrypts σ_i with attribute k_i to abCT_i by ABE, and outputs $\{\mathsf{abCT}_i\}$. In this construction, observe that the adversary cannot obtain illegitimate secret keys. Recall that in AB-MIFE for AWS, an ABE scheme for $\overline{\mathsf{P}}$ corresponds to ciphertext-policy ABE (CP-ABE) for ABPs, which was recently proposed by Lin and Luo [LL20b].

We observe that this security against legitimate vs. any keys in the context of AB-MIFE can be seen as generalization of security against complete vs. incomplete (or zero vs. multiple) queries in the context of MIFE [AGRW17]. Recall that incomplete queries refers to the case where an adversary does not make a ciphertext query for some inputs. Therefore, in the context of plain MIFE, all secret keys become illegitimate if the adversary makes incomplete queries and legitimate otherwise. On the other hand, in the context of AB-MIFE, whether each secret key become legitimate or illegitimate crucially depends on which attributes are queried in both ciphertext and secret-key queries, and thus the situation is much more complex. This is why we need an advanced primitive, namely, ABE to upgrade the security of AB-MIFE while MIFE secure against complete queries can be upgraded to that secure against incomplete queries using only symmetric key encryption.

Security under corruptions. The above transformation works only in the secret-key setting where the adversary cannot corrupt encryption keys. Intuitively, this limitation arises from the fact that there exist ABPs that never evaluate to 0 (we call such ABPs null ABPs). For the transformation to work in the corruption model, we require the underlying CP-ABE scheme to have the property that the adversary cannot decrypt ciphertexts for null ABPs even if it obtains the master secret key. However, in the only known CP-ABE scheme for ABPs by [LL20b], the master secret key has the ability to decrypt ciphertexts for null ABPs. Indeed, such a CP-ABE scheme implies witness encryption for NP relations verifiable in NC_1, and seems quite challenging to obtain from standard assumptions.

To circumvent this problem, we introduce wildcards for the access-controlling functionality similarly to [FFMV23]. In more detail, for the wildcard input $\star$ and all ABPs (including null ABPs) g, we always have $g(\star) = 0$. In this functionality, the adversary that corrupts i-th input can admissibly generate a ciphertext for slot i that satisfies any i-th predicate of secret keys, and the leakage of the master secret key of the CP-ABE scheme does not give any additional information to the adversary. As observed in [FFMV23], although multi-input ABE with corruptions in general implies witness encryption, their constructions do not yield witness encryption because of the use of wildcards. The same applies to our work as well.

To allow our AB-MIFE scheme to support wildcards, the underlying AB-FE scheme for AWSw/IP also needs to have the wildcard functionality. This modification is quite simple: just setting $v = 0$ (see step 1 above) in encryption with the wildcard attribute suffices.

Comparison with [NPP22]. Very recently, Nguyen, Phan and Pointcheval proposed an attribute-based MCFE scheme for inner product (see Table 3 for precise functionality). Their scheme is in the weaker MCFE model where each label can be used only once per input, and does not imply standard MIFE for the same function class. In [NPP22, Remark 16], they informally state that we can apply 1) the technique in [CDG+18b] to convert their scheme into the MCFE in the stronger notion and 2) All-or-Nothing Encapsulation [CDSG+20] to achieve security against incomplete queries. We believe that both claims are false.

Regarding item 1, as we discussed previously, the technique in [CDG+18b] to remove the one-time restriction requires ciphertext homomorphism of the underlying scheme. However, their underlying single client scheme is not ciphertext homomorphic, and thus how to use the technique in [CDG+18b] is unclear. Regarding item 2, as discussed above, in the context of the *attribute-based* setting, All-or-Nothing Encapsulation would be insufficient to achieve full-fledged security in the AB-MIFE setting, which can handle only the issue of complete vs. incomplete queries in the non-attribute-based setting. Hence, their result does not appear to imply AB-MIFE scheme as claimed.

Multi-Client FE for AWS. To construct MCFE for AWS, we follow the blueprint by [AGT21] where they construct an MCFE scheme for inner products from a function-hiding IPFE scheme. Roughly speaking, we replace the function-hiding IPFE scheme in their scheme with our FE scheme for AWSw/IP. However, following this approach leads to obstacles in the security proof, to handle which, we need to modify their blueprint. To see this, first consider the MCFE construction for AWS that is obtained by applying their blueprint straightforwardly to our setting. Let $H : \{0,1\}^* \to G_1$ be a hash function and aFE be a FE scheme for AWSw/IP. The scheme is given as follows:

Construction 3 (Candidate MCFE for AWS).

$\mathsf{Setup}(1^\lambda)$**:** It outputs $\mathsf{EK}_i = \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$ and $\mathsf{MSK} = \{\mathsf{aMSK}_i\}_i$.
$\mathsf{Enc}(\mathsf{EK}_i, \mathbf{v}_i, L)$**:** It outputs $\mathsf{CT}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{v}_i, [(v_L, 0)]_1))$ where $[v_L]_1 = H(L)$.
$\mathsf{KeyGen}(\mathsf{MSK}, (F_1, \dots, F_n))$**:** It outputs $\mathsf{SK} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where $r_1, \dots, r_n \leftarrow \mathbb{Z}_p$ s.t. $\sum r_i = 0$ and $\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (F_i, [(r_i, 0)]_2))$.
$\mathsf{Dec}(\mathsf{CT}_1, \dots, \mathsf{CT}_n, \mathsf{SK})$**:** It outputs $\sum_{i\in[n]} \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i) = [\sum F_i(\mathbf{v}_i)]_T$.

Let us try to prove the security of this MCFE candidate similarly to Construction 1. In what follows, we denote $\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{v}, [\mathbf{p}]_1))$ and $\mathsf{aKeyGen}(\mathsf{aMSK}_i, (F, [\mathbf{q}]_2))$ by $\mathsf{aCT}_i[\mathbf{v}, \mathbf{p}]$ and $\mathsf{aSK}_i[F, \mathbf{q}]$, respectively. To see why the security proof does not work in this construction, considering the simple case suffices where an adversary queries only one challenge ciphertext for each slot after which it makes secret-key queries adaptively. In the original game, the adversary is given $\mathsf{aCT}_i[\mathbf{v}_i^\beta, (v_L, 0)]$ for a challenge message $(\mathbf{v}_i^0, \mathbf{v}_i^1, L)$ and $\{\mathsf{aSK}_i[F_i, (r_i, 0)]\}_{i\in[n]}$ for a secret key of $(F_1, \dots, F_n)$. In the first hybrid, the ciphertext for slot i is changed to $\mathsf{aCT}_i[\mathbf{v}^0, (0, 1)]$ while all secret keys are changed to $\{\mathsf{aSK}_i[F_i, (r_i, v_L r_i + F_i(\mathbf{v}_i^\beta) - F_i(\mathbf{v}_i^0))]\}_{i\in[n]}$. The indistinguishability of the original game and the hybrid follows from the partially function-hiding security of aFE. The next step will be to change $[v_L r_i]_2$ to $[\widetilde{r}_i]_2$ where $\widetilde{r}_i$ is a random element in $\mathbb{Z}_p$ such that $\sum \widetilde{r}_i = 0$. If we can show this indistinguishability, $\widetilde{r}_i$ absorbs the term $F_i(\mathbf{v}_i^\beta) - F_i(\mathbf{v}_i^0)$ and we can conclude the proof, but this is not the case. This is because the adversary can compute $[v_L]_1$ by the hash function, and thus we cannot use the SXDH assumption in G_2.

We solve this by modifying Construction 3 as follows: Let $\mathsf{PRF} : \{0,1\}^* \to \mathbb{Z}_p$ be a pseudorandom function with a key space $\mathcal{K}$.

Construction 4 (MCFE for AWS).

$\mathsf{Setup}(1^\lambda)$**:** It chooses $\mathsf{K}_{i,j} \leftarrow \mathcal{K}$ for $i, j \in [n], i < j$, and sets $\mathsf{K}_{i,j} = \mathsf{K}_{j,i}$ for $j < i$. It outputs $\mathsf{EK}_i = (\mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda), \{\mathsf{K}_{i,j}\}_{i\neq j})$ for $i \in [n]$ and $\mathsf{MSK} = \{\mathsf{aMSK}_i\}_i$.
$\mathsf{Enc}(\mathsf{EK}_i, x_i, L)$**:** It outputs $\mathsf{CT}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (x, [(v_{L,i}, 0)]_1))$ where $v_{L,i} = \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L)$.
$\mathsf{KeyGen}(\mathsf{MSK}, (f_1, \dots, f_n))$**:** It outputs $\mathsf{SK} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where $r \leftarrow \mathbb{Z}_p$ and $\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (f_i, [(r, 0)]_2))$.
$\mathsf{Dec}(\mathsf{CT}_1, \dots, \mathsf{CT}_n, \mathsf{SK})$**:** It outputs $\sum_{i\in[n]} \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i) = [\sum f_i(x_i)]_T$.

This construction is inspired by the MIFE scheme in [AGRW17], in which the randomizing term v_i in ciphertexts are generated in the setup phase (not by a PRF). Note that this is enough for MIFE, but in our case we extend the technique to generate the term $v_{L,i}$ for each label on the fly via PRF to handle an exponentially large number of labels. Observe that $\sum_{i\in[n]} v_{L,i} = 0$ for all L and correctness holds. Such a usage of PRF in MCFE was first introduced by [ABG19], but again their MCFE construction requires ciphertext homomorphism and is not applicable to AWS functionality. This is why we devise a new construction based on DOT combining ideas from [AGRW17, ABG19].

In this construction, the above proof strategy works. At a high level, this is due to the following reasons:

- $\{v_{L,i}\}$ looks random with the constraint $\sum v_{L,i} = 0$ for each label if the PRF is secure.
- In contrast to Construction 3, the adversary cannot compute $v_{L,i}$ publicly.

In fact, Construction 4 is a secure MCFE scheme for AWS. For a detailed description, we refer the reader to Section 5.

Dynamic Decentralized FE for AWS. Given an MCFE scheme for AWS, we now convert it to DDFE using the template provided by [AGT21]. The high-level idea of the blueprint is to allow parties in the system to generate an independent MCFE instance in Construction 4 for each user set $\mathcal{U}$ by using a PRF on the fly. First, each party joins the system dynamically by generating a key K_i of a pseudorandom function (PRF) as a master secret key. In encryption and key generation for party set $\mathcal{U}$, party $i \in \mathcal{U}$ generates $\mathsf{aMSK}_{i,\mathcal{U}} = \mathsf{aSetup}(1^\lambda; \mathsf{PRF}^{\mathsf{K}_i}(\mathcal{U}))$, which is unique to $(i,\mathcal{U})$. For key generation of $(\{f_i\},\mathcal{U})$, party i computes a common random element $r_{i,\mathcal{U}} = H(\{f_i\},\mathcal{U})$ by a hash function and outputs $\mathsf{aSK}_{i,\mathcal{U}}$ as KeyGen in Construction 4. In encryption of $(x_i,\mathcal{U},L)$, party i generates $\mathsf{K}_{i,j}$ via non-interactive key exchange, and outputs $\mathsf{aCT}_{i,\mathcal{U}}$ in the same manner as Enc in Construction 4. Observe that $\mathsf{aCT}_{i,\mathcal{U}}/\{\mathsf{aSK}_{i,\mathcal{U}}\}_{i\in\mathcal{U}}$ is a valid ciphertext/secret key of the MCFE scheme in Construction 4 with respect to $\mathcal{U}$. For more details, please see Section 6.

We provide an overview of our constructions in Figure 2.

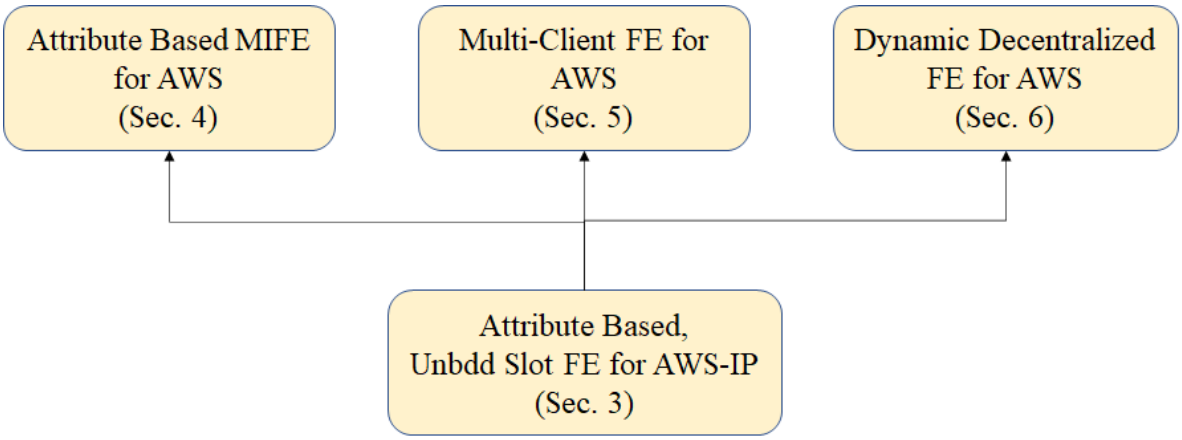

Fig 2. Outline of our constructions. For the implication to MCFE and DDFE, the underlying construction for AWSw/IP need not be attribute based.

2 Preliminaries

In this section, we give definitions needed for this paper.

Notations. We use $[n]$ to denote the set $\{1,\dots,n\}$. For vector $\mathbf{v}$, $\mathbf{v}[i]$ denotes the i-th element of $\mathbf{v}$. For vectors $\mathbf{v}_1,\dots,\mathbf{v}_n$, $(\mathbf{v}_1,\dots,\mathbf{v}_n)$ denotes the vector concatenation as row vectors *regardless of* whether each $\mathbf{v}_i$ is a row or column vector. For a matrix $\mathbf{A} = (a_{j,\ell})_{j,\ell}$ over $\mathbb{Z}_p$, $[\mathbf{A}]_i$ denotes a matrix over G_i whose (j,ℓ)-th entry is $g_i^{a_{j,\ell}}$, and we use this notation for vectors and scalars similarly. We use addition for the group operation in every group in bilinear groups. For vectors $\mathbf{a} \in \mathbb{Z}_p^n$ and $\mathbf{b} \in G^n$ where G is a cyclic group of order p, we abuse the notation of inner product and denote $\sum_{i\in[n]} \mathbf{a}[i][\mathbf{b}[i]]$ by $\langle \mathbf{a}, [\mathbf{b}] \rangle$. For a matrix $\mathbf{M} \in \mathbb{Z}_p^{a\times b}$ and vectors $\mathbf{a} \in \mathbb{Z}_p^a, \mathbf{b} \in \mathbb{Z}_p^b$, we denote a vector $\mathbf{m}$ such that $\langle \mathbf{a}\otimes\mathbf{b}, \mathbf{m}\rangle = \mathbf{a}^\top\mathbf{M}\mathbf{b}$ by $\mathsf{vec}(\mathbf{M})$.

2.1 Computation Models

Definition 2.1 (Arithmetic Branching Programs (ABPs)). An arithmetic branching program $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p$ is defined by a prime p, a directed acyclic graph (V,E), two special vertices $v_0, v_1 \in V$, and a labeling function $\sigma : E \to \mathcal{F}^{\mathsf{Affine}}$, where $\mathcal{F}^{\mathsf{Affine}}$ consists of all affine functions $g : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p$. The

size of f is the number of vertices $|V|$. Given an input $\mathbf{x} \in \mathbb{Z}_p^{n_0}$ to the ABP, we can assign a $\mathbb{Z}_p$ element to edge $e \in E$ by $\sigma(e)(\mathbf{x})$. Let P be the set of all paths from v_0 to v_1. Each element in P can be represented by a subset of E. The output of the ABP on input $\mathbf{x}$ is defined as $\sum_{E' \in P} \prod_{e \in E'} \sigma(e)(\mathbf{x})$. We can extend the definition of ABPs for functions $f : \mathbb{Z}_p^{n_0} \to \mathbb{Z}_p^{n_1}$ by evaluating each output in a coordinate-wise manner and denote such a function class by $\mathcal{F}_{n_0,n_1}^{\mathsf{ABP}}$.

Note that we can convert any boolean formula, boolean branching program or arithmetic formula to an arithmetic branching program with a constant blow-up in the representation size. Thus, ABPs are a stronger computational model than all of the above.

2.2 Basic Cryptographic Notions

Definition 2.2 (Bilinear Groups). Let $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of bilinear groups. Bilinear groups $\mathbb{G}_\lambda$=$(p, G_1, G_2, G_T, g_1, g_2, e)$ are specified by a prime p, cyclic groups G_1, G_2, G_T of order p, generators g_1 and g_2 of G_1 and G_2 respectively, and a bilinear map $e : G_1 \times G_2 \to G_T$, which has two properties.

- (Bilinearity): $\forall h_1 \in G_1, h_2 \in G_2, a, b \in \mathbb{Z}_p, e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
- (Non-degeneracy): For g_1 and g_2, $g_T = e(g_1, g_2)$ is a generator of G_T.

In what follows, we omit the index λ from $\mathbb{G}_\lambda$ and abuse notation by denoting a family of bilinear groups $\{\mathbb{G}_\lambda\}_{\lambda \in \mathbb{N}}$ also by $\mathbb{G}$ if it is clear in the context.

Definition 2.3 ($\mathcal{D}_{j,k}$-MDDH Assumption [EHK+17]). Let $\{\mathbb{G}\}$ be a family of bilinear groups. For $j > k$, let $\mathcal{D}_{j,k}$ be a matrix distribution over matrices in $\mathbb{Z}_p^{j \times k}$, which outputs a full-rank matrix with overwhelming probability. We can assume that, wlog, the first k rows of a matrix chosen from $\mathcal{D}_{j,k}$ form an invertible matrix. We consider the following distribution: $\mathbf{A} \leftarrow \mathcal{D}_{j,k}$, $\mathbf{m} \leftarrow \mathbb{Z}_p^k$, $\mathbf{k}_0 = \mathbf{A}\mathbf{m}$, $\mathbf{k}_1 \leftarrow \mathbb{Z}_p^j$, $P_{i,\beta} = (\mathbb{G}, [\mathbf{A}]_i, [\mathbf{k}_\beta]_i)$. We say that the $\mathcal{D}_{j,k}$-MDDH assumption holds with respect to $\{\mathbb{G}\}$ if, for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_{j,k}\text{-}\mathsf{MDDH}} = \max_{i \in \{1,2\}} |\Pr[1 \leftarrow \mathcal{A}(P_{i,0})] - \Pr[1 \leftarrow \mathcal{A}(P_{i,1})]| \leq \mathsf{negl}(\lambda).$$

In what follows, we denote $\mathcal{D}_{k+1,k}$ by $\mathcal{D}_k$. Note that the well-known k-Lin assumption can be captured as the $\mathcal{D}_k$-MDDH assumption.

Uniform Distribution. Let $\mathcal{U}_{j,k}$ be a uniform distribution over $\mathbb{Z}_p^{j \times k}$. Then, the following holds with tight reductions: $\mathcal{D}_k$-MDDH $\Rightarrow \mathcal{U}_k$-MDDH $\Rightarrow \mathcal{U}_{j,k}$-MDDH. We denote $\mathcal{D}_k$-MDDH by MDDH_k.

Random Self-Reducibility. We can obtain arbitrarily many instances of the $\mathcal{D}_{j,k}$-MDDH problem from a single instance. For any $n \in \mathbb{N}$, we define the following distribution: $\mathbf{A} \leftarrow \mathcal{D}_{j,k}$, $\mathbf{M} \leftarrow \mathbb{Z}_p^{k \times n}$, $\mathbf{K}_0 = \mathbf{A}\mathbf{M}$, $\mathbf{K}_1 \leftarrow \mathbb{Z}_p^{j \times n}$, $P_{i,\beta} = (\mathbb{G}, [\mathbf{A}]_i, [\mathbf{K}_\beta]_i)$. The n-fold $\mathcal{D}_{j,k}$-MDDH assumption is similarly defined to the $\mathcal{D}_{j,k}$-MDDH assumption. Then, the n-fold $\mathcal{D}_{j,k}$-MDDH assumption is implied by the $\mathcal{D}_{j,k}$-MDDH assumption with security loss of $\min\{n, j-k\}$.

Definition 2.4 (Pseudorandom functions (PRFs)). A pseudorandom function (PRF) family $\mathcal{F} = \{\mathsf{PRF}^{\mathsf{K}}\}_{\mathsf{K} \in \mathcal{K}}$ with a key space $\mathcal{K}$, a domain $\mathcal{X}$, and a range $\mathcal{Y}$ is a function family that consists of functions $\mathsf{PRF}^{\mathsf{K}} : \mathcal{X} \to \mathcal{Y}$. Let $\mathcal{R}$ be a set of functions consisting of all functions whose domain and range are $\mathcal{X}$ and $\mathcal{Y}$ respectively. A PRF family $\mathcal{F}$ is said to be secure if for any PPT adversary $\mathcal{A}$, the following condition holds,

$$|\Pr[\mathcal{A}^{\mathsf{PRF}^{\mathsf{K}}(\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{A}^{R(\cdot)}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda),$$

where $K \leftarrow \mathcal{K}$ and $R \leftarrow \mathcal{R}$.

Definition 2.5 (Secret Sharing Scheme). A (n out of n) secret sharing scheme consists of Share and Rec.

$\mathsf{Share}(s, n)$**:** It takes a secret $s \in \{0,1\}^m$ and a number of shares n and outputs shares $\sigma_1, \ldots, \sigma_n \in \{0,1\}^m$.
$\mathsf{Rec}(\sigma_1, \ldots, \sigma_n)$**:** It takes shares $\sigma_1, \ldots, \sigma_n \in \{0,1\}^m$ and outputs a bit string s'.

A secret sharing scheme has two properties.

Correctness: For all $n, m \in \mathbb{N}, s \in \{0,1\}^m$,

$$\Pr[\mathsf{Rec}(\sigma_1, \ldots, \sigma_n) = s : \sigma_1, \ldots, \sigma_n \leftarrow \mathsf{Share}(s, n)] = 1.$$

Security: For all $n, m \in \mathbb{N}, s \in \{0,1\}^m, S \subsetneq [n]$, the following distributions are identical:

$$\{\{\sigma_i\}_{i \in S} : \sigma_1, \ldots, \sigma_n \leftarrow \mathsf{Share}(s, n)\} \quad \text{and} \quad \{\{\sigma_i\}_{i \in S} : \sigma_1, \ldots, \sigma_n \leftarrow \{0,1\}^m\}$$

Definition 2.6 (Non-interactive key exchange (NIKE)). A NIKE scheme for shared key space $\mathcal{K}$ consists of the three algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{PP}$**:** It takes a security parameter 1^λ and outputs a public parameter PP.
$\mathsf{KeyGen}(\mathsf{PP}) \to (\mathsf{PK}, \mathsf{SK})$**:** It takes PP and outputs a public key PK and the corresponding secret key SK.
$\mathsf{SharedKey}(\mathsf{PK}, \mathsf{SK}) \to \mathsf{K}$**:** It takes PK and SK and deterministically outputs a shared key $\mathsf{K} \in \mathcal{K}$.

Correctness. A NIKE scheme is correct if, for all $\lambda \in \mathbb{N}$, we have

$$\Pr\left[\mathsf{K}_{i,j} = \mathsf{K}_{j,i} : \begin{array}{l} \mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda) \\ (\mathsf{PK}_i, \mathsf{SK}_i), (\mathsf{PK}_j, \mathsf{SK}_j) \leftarrow \mathsf{KeyGen}(\mathsf{PP}) \\ \mathsf{K}_{i,j} = \mathsf{SharedKey}(\mathsf{PK}_i, \mathsf{SK}_j) \\ \mathsf{K}_{j,i} = \mathsf{SharedKey}(\mathsf{PK}_j, \mathsf{SK}_i) \end{array}\right] = 1.$$

Security. We say a NIKE scheme is IND-secure if, for all stateful PPT adversaries $\mathcal{A}$, we have

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\},\ \mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathcal{S} \leftarrow \mathcal{A}(\mathsf{PP}) \\ (\mathsf{PK}_i, \mathsf{SK}_i) \leftarrow \mathsf{KeyGen}(\mathsf{PP}) \\ \mathcal{CS}, (i', j') \leftarrow \mathcal{A}(\{\mathsf{PK}_i\}_{i \in \mathcal{S}}) \text{ where } i', j' \in \mathcal{S} \backslash \mathcal{CS} \text{ and } i' \neq j' \\ \mathsf{K}^0_{i',j'} = \mathsf{SharedKey}(\mathsf{PK}_{i'}, \mathsf{SK}_{j'}),\ \mathsf{K}^1_{i',j'} \leftarrow \mathcal{K} \\ \beta' \leftarrow \mathcal{A}(\{\mathsf{SK}_i\}_{i \in \mathcal{CS}}, \mathsf{K}^\beta_{i',j'}) \end{array}\right] \leq 1/2 + \mathsf{negl}(\lambda).$$

Definition 2.7 (Partial Garbling Scheme for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$). We use the following partial garbling scheme for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ [IW14] (please see Definition 2.1) for the construction of our FE schemes. A partial garbling scheme for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ consists of the four algorithms. Note that lgen and rec are deterministic algorithms while pgb and pgb^* are probabilistic algorithms.

$\mathsf{lgen}(f)$**:** It takes $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$ and outputs $\mathbf{L}_1, \ldots, \mathbf{L}_t \in \mathbb{Z}_p^{(n_0+1)\times(t-1)}$ where t depends on f.
$\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t})$**:** Let $\mathbf{x}'^\top = (\mathbf{x}, 1)$. It takes $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$. It then outputs

$$(\mathbf{x}'^\top \mathbf{L}_1 \mathbf{t}, \ldots, \mathbf{x}'^\top \mathbf{L}_s \mathbf{t}, \mathbf{z}[1] + \mathbf{x}'^\top \mathbf{L}_{s+1} \mathbf{t}, \ldots, \mathbf{z}[n_1] + \mathbf{x}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $s = t - n_1$ and $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.
$\mathsf{pgb}^*(f, \mathbf{x}, \mu; \mathbf{t})$**:** It takes $\mu \in \mathbb{Z}_p$ and $f, \mathbf{x}, \mathbf{t}$ as above and outputs

$$(\mathbf{x}'^\top \mathbf{L}_1 \mathbf{t} + \mu, \mathbf{x}'^\top \mathbf{L}_2 \mathbf{t}, \ldots, \mathbf{x}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

$\mathsf{rec}(f, \mathbf{x})$**:** It takes $f, \mathbf{x} \in \mathbb{Z}_p^{n_0}$ and outputs $\mathbf{d}_{f,\mathbf{x}} \in \mathbb{Z}_p^t$.

The concrete description of $\mathsf{lgen}, \mathsf{rec}$ that satisfy the following properties is found in [AGW20, Appendix A]. We slightly modify the format of the output of lgen from [AGW20] for convenience in our construction, but note that they are essentially the same.

Correctness. The garbling scheme is correct if for all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}, \mathbf{t} \in \mathbb{Z}_p^{t-1}$, we have

$$\langle \mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}), \mathsf{rec}(f, \mathbf{x}) \rangle = \langle f(\mathbf{x}), \mathbf{z} \rangle.$$

Security. The garbling scheme is secure if for all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}$, the following distributions are statistically close:

$$\mathsf{pgb}(f, \mathbf{x}, \mathbf{z}; \mathbf{t}) \quad \text{and} \quad \mathsf{pgb}^*(f, \mathbf{x}, \langle f(\mathbf{x}), \mathbf{z} \rangle; \mathbf{t})$$

where the random tape is chosen over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$.

Extension of Partial Garbling Scheme. We can construct an additional partial garbling algorithm pgb^+ with the following properties [AGW20, Appendix A].

$\mathsf{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t})$**:** Let $\mathbf{x}'^\top = (\mathbf{x}, 1)$. It takes $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}, \delta \in \mathbb{Z}_p$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$. It then outputs

$$(\mathbf{x}'^\top \mathbf{L}_1 \mathbf{t} + \boxed{\delta}, \mathbf{x}'^\top \mathbf{L}_2 \mathbf{t}, \ldots, \mathbf{x}'^\top \mathbf{L}_s \mathbf{t}, \mathbf{z}[1] + \mathbf{x}'^\top \mathbf{L}_{s+1} \mathbf{t}, \ldots, \mathbf{z}[n_1] + \mathbf{x}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $s = t - n_1$ and $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

Correctness. For all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}, \mathbf{t} \in \mathbb{Z}_p^{t-1}$, we have

$$\langle \mathsf{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}), \mathsf{rec}(f, \mathbf{x}) \rangle = \langle f(\mathbf{x}), \mathbf{z} \rangle + \delta.$$

Security. For all $f \in \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}, \mathbf{x} \in \mathbb{Z}_p^{n_0}, \mathbf{z} \in \mathbb{Z}_p^{n_1}$, the following distributions are statistically close:

$$\mathsf{pgb}^+(f, \mathbf{x}, \mathbf{z}, \delta; \mathbf{t}) \quad \text{and} \quad \mathsf{pgb}^*(f, \mathbf{x}, \langle f(\mathbf{x}), \mathbf{z} \rangle + \delta; \mathbf{t})$$

where the random tape is chosen over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$.

Linearlity. Observe that pgb^+ is affine in $\mathbf{z}[1]$, $\mathbf{t}$, δ, and pgb^* is affine in μ.

2.3 Variants of Functional Encryption

Definition 2.8 (Attribute-Based Encryption (ABE)). Let $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ be a predicate where $\mathcal{X}$ and $\mathcal{Y}$ denote ciphertext-attribute and key-attribute spaces. An attribute-based encryption (ABE) scheme for a predicate family P consists of four algorithms:

$\mathsf{Setup}(1^\lambda)$**:** It takes a security parameter 1^λ, and outputs a public key PK and a master secret key MSK. The other algorithms implicitly take PK.

$\mathsf{Enc}(x, M)$**:** It takes PK, an attribute $x \in \mathcal{X}$ and a message $M \in \mathcal{M}$ as inputs, and outputs a ciphertext CT. (Note that we let $\mathcal{M}$ be specified in PK.)

$\mathsf{KeyGen}(\mathsf{MSK}, y)$**:** It takes $\mathsf{PK}, \mathsf{MSK}$, and an attribute $y \in \mathcal{Y}$ as inputs, and outputs a secret key SK.

$\mathsf{Dec}(\mathsf{CT}_x, \mathsf{SK}_y)$**:** It takes PK, CT and SK as inputs, and outputs a message M' or a symbol $\perp$.

Correctness. An ABE scheme is correct if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $x \in \mathcal{X}$, $y \in \mathcal{Y}$ such that $\mathsf{P}(x, y) = 1$, and $M \in \mathcal{M}$, we have

$$\mathsf{Pr}\left[M = M' : \begin{array}{l} (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, \kappa) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(x, M) \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, y) \\ M' = \mathsf{Dec}(\mathsf{CT}, \mathsf{SK}) \end{array} \right] = 1.$$

Security. An ABE scheme is selectively secure in the multi-challenge setting if it satisfies the following condition. That is, the advantage of $\mathcal{A}$ defined as follows is negligible in λ for all stateful PPT adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{ABE}}_{\mathcal{A}}(\lambda) = \Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, \kappa) \\ \{x^j, M_0^k, M_1^j\}_{j\in[q_c]} \leftarrow \mathcal{A}(\mathsf{PK}) \\ \mathsf{CT}^j \leftarrow \mathsf{Enc}(x^j, M_\beta^j) \ \text{ for } j \in [q_c] \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\{\mathsf{CT}^j\}_{j\in[q_c]}) \end{array}\right] - \frac{1}{2}$$

where all $\{y^\ell\}_{\ell\in[q_k]}$ on which $\mathcal{A}$ queries KeyGen must satisfy $\mathsf{P}(x^j, y^\ell) = 0$.

Definition 2.9 (Secret-Key Functional Encryption). Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X} \to \mathcal{Z}$. A secret-key functional encryption (SK-FE) scheme for $\mathcal{F}$ consists of four algorithms.

$\mathsf{Setup}(1^\lambda)$**:** It takes a security parameter 1^λ and outputs a public parameter PP, and a master secret key MSK. The other algorithms implicitly take PP.
$\mathsf{Enc}(\mathsf{MSK}, x)$**:** It takes MSK and $x \in \mathcal{X}$ and outputs a ciphertext CT.
$\mathsf{KeyGen}(\mathsf{MSK}, f)$**:** It takes MSK and $f \in \mathcal{F}$, and outputs a secret key SK.
$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** It takes CT and SK, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. An SK-FE scheme is correct if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $x \in \mathcal{X}$, $f \in \mathcal{F}$, we have

$$\Pr\left[f(x) = \mathsf{Dec}(\mathsf{CT}, \mathsf{SK}) : \begin{array}{l} (\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x) \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \end{array}\right] = 1.$$

Security. We consider the case where each $x \in \mathcal{X}$ consists of a public part x_{pub} and a private part x_{priv}, i.e., $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})$, and each $f \in \mathcal{F}$ consists of a public part f_{pub} and a private part f_{priv}, i.e., $f = (f_{\mathsf{pub}}, f_{\mathsf{priv}})$. An SK-FE scheme is selectively partially function-hiding if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QEnc}^\beta(),\mathsf{QKeyGen}^\beta()}(\mathsf{PP}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $x^{j,\beta} = (x^j_{\mathsf{pub}}, x^{j,\beta}_{\mathsf{priv}})$, $f^\beta = (f_{\mathsf{pub}}, f^\beta_{\mathsf{priv}})$, $\mathsf{QEnc}^\beta(x^0, x^1)$ returns $\mathsf{Enc}(\mathsf{MSK}, x^\beta)$, and $\mathsf{QKeyGen}^\beta(f^0, f^1)$ returns $\mathsf{KeyGen}(\mathsf{MSK}, f^\beta)$. The admissible adversary's queries must satisfy the following condition:

1. $\mathcal{A}$ cannot query QEnc^β after querying $\mathsf{QKeyGen}^\beta$ even once.
2. If (x^0, x^1) is included in the query to QEnc^β and (f^0, f^1) is queried to $\mathsf{QKeyGen}^\beta$, then $f^0(x^0) = f^1(x^1)$.

3 Attribute-Based FE for Attribute-Weighted Sums with Inner Product

In this section, we present an attribute-based FE for attribute-weighted sums with inner product (AB-FE for AWSw/IP). In Appendix B, we show how it can be captured using the notation of MPFE. We will need the following definitions.

Definition 3.1 (Inner Product Functional Encryption). Inner product functional encryption (IPFE) is a class of secret-key functional encryption (SK-FE) that supports the following functionality. Let $\mathbb{G}$ be bilinear groups. Let $\mathcal{X} = G_1^m$ be a message space. Let $\mathcal{F} = G_2^m$ be a family of functions, where $f = [\mathbf{c}]_2 \in \mathcal{F}$ represents the function $f : \mathcal{X} \to G_T$ defined as $f([\mathbf{x}]_1) = [\langle \mathbf{x}, \mathbf{c} \rangle]_T$ where $\mathbf{x}, \mathbf{c} \in \mathbb{Z}_p^m$ are both private inputs.

A function-hiding IPFE scheme can be constructed from the MDDH assumption [Tom19, Appendix A].

Definition 3.2 (FE for AWSw/IP). An FE scheme for attribute-weighted sums with inner product (AWSw/IP) is a class of SK-FE that supports the following functionality. Let $\mathbb{G}$ be bilinear groups of order p. Let $\mathcal{X} = \bigcup_{i \in \mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i \times G_1^m$ be a message space. Let $\mathcal{F} = \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1} \times G_2^m$ (see Definition 2.1 for $\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1}$) be a family of functions, where $f' = (f, [\mathbf{q}]_2) \in \mathcal{F}$ represents the function $f' : \mathcal{X} \to G_T$ defined as

$$f'((\{\mathbf{x}_i, \mathbf{z}_i\}_{i \in [N]}, [\mathbf{p}]_1)) = [\sum_{i \in [N]} \langle f(\mathbf{x}_i), \mathbf{z}_i \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T$$

where $\{\mathbf{x}_i\}, f$ are public elements while $\{\mathbf{z}_i\}, [\mathbf{p}]_1, [\mathbf{q}]_2$ are private elements.

Definition 3.3 (AB-FE for AWSw/IP). An attribute-based FE scheme for attribute-weighted sums with inner product (AB-FE for AWSw/IP) is a class of SK-FE that supports the following functionality. Let $\mathbb{G}$ be bilinear groups. Let $\mathcal{X} = (\mathbb{Z}_p^{n_0'} \cup \{\star\}) \times \bigcup_{i \in \mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i \times G_1^m$ be a message space. Let $\mathcal{F} = \mathcal{F}^{\mathsf{ABP}}_{n_0',1} \times \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1} \times G_2^m$ be a family of functions, where $f = (g, h, [\mathbf{q}]_2) \in \mathcal{F}$ represents the function $f : \mathcal{X} \to G_T$ defined as

$$f((\mathbf{y}, \{\mathbf{x}_j, \mathbf{z}_j\}_{j \in [N]}, [\mathbf{p}]_1)) = \begin{cases} [\sum_{j \in [N]} \langle h(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T & g(\mathbf{y}) = 0 \vee \mathbf{y} = \star \\ \bot & g(\mathbf{y}) \neq 0 \end{cases}$$

where $\mathbf{y}, \{\mathbf{x}_i\}, g, h$ are public elements while $\{\mathbf{z}_i\}, [\mathbf{p}]_1, [\mathbf{q}]_2$ are private elements. For notational convenience, we define $g(\star) = 0$ for all ABPs g (even for ABPs g such that $g(\mathbf{y}) \neq 0$ for all $\mathbf{y} \in \mathbb{Z}_p^{n_0'}$).

Remark 3.1. As explained in the introduction, we need the wildcard functionality to make our AB-MIFE scheme secure in the corruption model. This is why we define the functionality of AB-FE for AWSw/IP such that it also supports wildcards, which we will use to construct our our AB-MIFE scheme as a building block.

3.1 Construction

Let k be the parameter for the MDDH$_k$ assumption. Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme with the vector length being $k(n_0' + n_0 + 3) + n_1 + 2m + 2$. The last $m + 2$ elements are used for only the security proof. Let $(\mathsf{lgen}, \mathsf{pgb}, \mathsf{pgb}^+, \mathsf{pgb}^*, \mathsf{rec})$ be a partially garbling scheme for ABPs (Definition 2.7). Our AB-FE scheme for AWSw/IP is given in Figure 3.

Correctness and Security. In decryption, due to the correctness of iFE, we have

$$\mathbf{d}_j = \begin{cases} \mathsf{pgb}^+(\phi, (\mathbf{y}, \mathbf{x}_j), (\langle \mathbf{s}, \mathbf{v} \rangle, \mathbf{z}_j), \langle \mathbf{r}, \mathbf{w}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle; \mathbf{T}\mathbf{u}_j) & (j = 1) \\ \mathsf{pgb}^+(\phi, (\mathbf{y}, \mathbf{x}_j), (\quad 0, \quad \mathbf{z}_j), \langle \mathbf{r}, \mathbf{w}_j \rangle \qquad\quad ; \mathbf{T}\mathbf{u}_j) & (j > 1) \end{cases}$$

where $\mathbf{v} = \mathbf{0}$ if $\mathbf{y}' = \star$. Thanks to the correctness of the partial garbling scheme, we have

$$\langle \mathbf{d}_j, \mathsf{rec}(\phi, (\mathbf{y}, \mathbf{x}_j)) \rangle = \begin{cases} \langle \mathbf{s}, \mathbf{v} \rangle g(\mathbf{y}) + \langle h(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{r}, \mathbf{w}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle & (j = 1) \\ \langle h(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{r}, \mathbf{w}_j \rangle & (j > 1) \end{cases}$$

In the above, we use the fact that $\phi((\mathbf{y}, \mathbf{x}_j)) = (g(\mathbf{y}), h(\mathbf{x}_j)) \in \mathbb{Z}_p^{1+n_1}$. Hence $d = \sum_{j \in [N]} \langle h(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle$ if $g(\mathbf{y}') = 0$, since $\sum_{j \in [N]} \langle \mathbf{r}, \mathbf{w}_j \rangle = 0$.

$\mathsf{Setup}(1^\lambda)$**:** It runs $\mathsf{iPP}, \mathsf{iMSK} \leftarrow \mathsf{iSetup}(1^\lambda)$ and outputs $(\mathsf{PP}, \mathsf{MSK}) = (\mathsf{iPP}, \mathsf{iMSK})$.

$\mathsf{Enc}(\mathsf{MSK}, (\mathbf{y}', \{\mathbf{x}_j, \mathbf{z}_j\}_{j\in[N]}, [\mathbf{p}]_1))$**:** It samples $\mathbf{u}_1, \ldots, \mathbf{u}_N, \mathbf{w}_1, \ldots, \mathbf{w}_{N-1} \leftarrow \mathbb{Z}_p^k$ and sets $\mathbf{w}_N = -\sum_{j\in[N-1]} \mathbf{w}_j$. If $\mathbf{y}' = \star$, it sets $\mathbf{y} = 0^{n_0'}$ and $\mathbf{v} = 0^k$, otherwise it sets $\mathbf{y} = \mathbf{y}'$ and $\mathbf{v} \leftarrow \mathbb{Z}_p^k$. Then, it defines

$$\boldsymbol{\chi}_j^\top = (\mathbf{y}, \mathbf{x}_j, 1), \quad \mathbf{X}_j = \begin{cases} (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j, \mathbf{w}_j, \mathbf{v},\ \mathbf{p},\ 0^{m+2}) & (j=1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j, \mathbf{w}_j, 0^k, 0^m, 0^{m+2}) & (j>1) \end{cases}$$

and computes $\mathsf{iCT}_j \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{X}_j]_1)$ for all $j \in [N]$. It outputs $\mathsf{CT} = (\mathbf{y}, \{\mathbf{x}_j, \mathsf{iCT}_j\}_{j\in[N]})$.

$\mathsf{KeyGen}(\mathsf{MSK}, (g, h, [\mathbf{q}]_2))$**:** It samples $\mathbf{r}, \mathbf{s} \leftarrow \mathbb{Z}_p^k$ and defines an ABP $\phi : \mathbb{Z}_p^{n_0'+n_0} \to \mathbb{Z}_p^{1+n_1}$ as $\phi((\mathbf{y}, \mathbf{x})) = (g(\mathbf{y}), h(\mathbf{x}))$ for $\mathbf{y} \in \mathbb{Z}_p^{n_0'}, \mathbf{x} \in \mathbb{Z}_p^{n_0}$. It computes $\mathbf{L}_1, \ldots, \mathbf{L}_t \leftarrow \mathsf{lgen}(\phi)$ and $\mathbf{T} \leftarrow \mathbb{Z}_p^{(t-1)\times k}$ and defines

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q},\ 0^{m+2}) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m, 0^{m+2}) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m, 0^{m+2}) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^{m+2}) & & (s+1<j) \end{cases}$$

where s is the parameter of the partial garbling scheme defined in Definition 2.7. It computes $\mathsf{iSK}_j \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\mathbf{Y}_j]_2)$ for all $j \in [t]$ and outputs $\mathsf{SK} = (\phi, \{\mathsf{iSK}_j\}_{j\in[t]})$.

$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** It parse CT, SK as $(\mathbf{y}, \{\mathbf{x}_j, \mathsf{iCT}_j\}_{j\in[N]})$ and $(\phi, \{\mathsf{iSK}_j\}_{j\in[t]})$, respectively. It outputs $\perp$ if $g(\mathbf{y}') \neq 0$. Otherwise, it computes $[d_{j,\ell}]_T = \mathsf{iDec}(\mathsf{iCT}_j, \mathsf{iSK}_\ell)$ for $j \in [N], \ell \in [t]$ and outputs

$$[d]_T = \sum_{j\in[N]} \langle [\mathbf{d}_j]_T, \mathsf{rec}(\phi, (\mathbf{y}, \mathbf{x}_j)) \rangle.$$

where $\mathbf{d}_j = (d_{j,1}, \ldots, d_{j,t})$.

Fig 3. Attribute-Based FE for AWSw/IP

Remark 3.2. A partially-hiding FE scheme for AWSw/IP can be obtained from a partially-hiding AB-FE scheme for AWSw/IP by setting $n_0' = 0$ and g as the constant function that outputs 0.

We argue security via the following theorem.

Theorem 3.1. *If* iFE *is function-hiding, the partial garbling scheme is secure, and the* MDDH_k *assumption holds in* $\mathbb{G}$*, then the proposed AB-FE scheme for AWSw/IP is partially function-hiding as per Definition 2.9.*

Proof. We prove the theorem via a series of hybrid games H_ℓ^β for $\ell \in [q_c]$ where q_c is the number of ciphertext queries by the adversary. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta \approx_c \cdots \approx_c \mathsf{H}_{q_c}^\beta \approx_c \mathsf{H}_f^\beta$, where H_s^β for $\beta \in \{0,1\}$ is the original security game. Intuitively, in H_ℓ^β, we program the vectors $\mathbf{X}_j$ and $\mathbf{Y}_j$ in the ciphertexts and secret keys queried by the adversary such that the challenge ciphertexts in the first ℓ queries decrypt to $\sum\langle h(\mathbf{x}_j), \mathbf{z}_j^0\rangle + \langle \mathbf{p}^0, \mathbf{q}^0\rangle$ while the rest of ciphertexts decrypts to $\sum\langle h(\mathbf{x}_j), \mathbf{z}_j^\beta\rangle + \langle \mathbf{p}^\beta, \mathbf{q}^\beta\rangle$. Then, in the last hybrid, the adversary obtains no information about β, and we can conclude the proof.

Recall that in H_s^β the challenger replies

$$\mathsf{Enc}(\mathsf{MSK}, (\mathbf{y}', \{\mathbf{x}_j, \mathbf{z}_j^\beta\}_{j\in[N]}, [\mathbf{p}^\beta]_1)) \text{ for } \mathsf{QEnc}^\beta(\mathbf{y}', \{\mathbf{x}_j, \mathbf{z}_j^0, \mathbf{z}_j^1\}_{j\in[N]}, [\mathbf{p}^0]_1, [\mathbf{p}^1]_1)$$
$$\mathsf{KeyGen}(\mathsf{MSK}, (g, h, [\mathbf{q}^\beta]_2)) \text{ for } \mathsf{QKeyGen}^\beta(g, h, [\mathbf{q}^0]_2, [\mathbf{q}^1]_2)$$

where Enc and KeyGen work as specified in Fig 3. The hybrid H_ℓ^β is the same as H_s^β except the way of defining $\mathbf{X}_j$ in Enc and $\mathbf{Y}_j$ in KeyGen in the replies for ciphertext and secret-key queries. Specifically,

$\mathbf{X}_j$ in the ℓ'-th ciphertext query is defined as

$$\begin{cases} \mathbf{X}_j^{\ell'} = \begin{cases} (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \underline{\mathbf{z}_j^0}, \mathbf{w}_j, \mathbf{v},\ 0^m, \underline{\mathbf{p}^0}, 0^2) & (j=1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \underline{\mathbf{z}_j^0}, \mathbf{w}_j, 0^k, 0^m,\ 0^{m+2}) & (j>1) \end{cases} & (\ell' \leq \ell) \\ \mathbf{X}_j^{\ell'} = \begin{cases} (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \underline{\mathbf{z}_j^\beta}, \mathbf{w}_j, \mathbf{v},\ \mathbf{p}^\beta,\ 0^{m+2}) & (j=1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \underline{\mathbf{z}_j^\beta}, \mathbf{w}_j, 0^k, 0^m,\ 0^{m+2}) & (j>1) \end{cases} & (\ell' > \ell) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^\beta, \underline{\mathbf{q}^0}, 0^2) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m, 0^{m+2}) & (1<j\leq s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m, 0^{m+2}) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m, 0^{m+2}) & (s+1<j) \end{cases}$$

The hybrid H_f^β is the same as $\mathsf{H}_{q_c}^\beta$ except that $\mathbf{Y}_1$ for all queries are defined as

$$\mathbf{Y}_1 = (\mathsf{vec}(\mathbf{L}_1\mathbf{T}), 0^{n_1}, \mathbf{r}, 0^k, \underline{0^m}, \mathbf{q}^0, 0^2)$$

Note that the advantage of the adversary is 0 in H_f^β since it does not obtain the information of β. Hence, the theorem holds from Lemmata 3.1 and 3.2. □

Lemma 3.1. $\mathsf{H}_{q_c}^\beta \approx_c \mathsf{H}_f^\beta$ *if* iFE *is function-hiding.*

Proof. Observe that in $\mathsf{H}_{q_c}^\beta$, $\mathbf{X}_j$ is defined as

$$\mathbf{X}_j = \begin{cases} (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j^0, \mathbf{w}_j, \mathbf{v},\ 0^m, \mathbf{p}^0, 0^2) & (j=1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j^0, \mathbf{w}_j, 0^k, 0^m,\ 0^{m+2}) & (j>1) \end{cases}$$

for all queries to QEnc. Therefore, for all queries to QEnc and QKeyGen, we have

$$\langle \mathbf{X}_j, \mathbf{Y}_1 \rangle = \boldsymbol{\chi}_j^\top \mathbf{L}_1 \mathbf{T} \mathbf{u}_j + \langle \mathbf{w}_j, \mathbf{r} \rangle + \langle \mathbf{p}^0, \mathbf{q}^0 \rangle$$

in *both* $\mathsf{H}_{q_c}^\beta$ and H_f^β. Hence, the indistinguishability of $\mathsf{H}_{q_c}^\beta$ and H_f^β readily follows from the function-hiding security of iFE. □

Lemma 3.2. *Let* $\mathsf{H}_0^\beta = \mathsf{H}_s^\beta$. *For all* $\ell \in [q_c]$, *we have* $\mathsf{H}_{\ell-1}^\beta \approx_c \mathsf{H}_\ell^\beta$.

Proof. What this lemma asserts is that for the ℓ-th ciphertext CT^ℓ and any secret key SK, the cases where decryption of these reveals $\gamma^\beta = \sum \langle h(\mathbf{x}^\ell), \mathbf{z}^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle$ or $\perp$ and where it reveals $\gamma^0 = \sum \langle h(\mathbf{x}^\ell), \mathbf{z}^{\ell,0} \rangle + \langle \mathbf{p}^{\ell,0}, \mathbf{q}^0 \rangle$ or $\perp$ are indistinguishable. Note that $\gamma^\beta = \gamma^0 = \gamma$ due to the query condition. As in [AGW20], our goal is the hybrid where CT^ℓ is simulatable without $\mathbf{z}^\beta, \mathbf{p}^{\ell,\beta}$, and SK is simulatable from γ. At this point, we can use the equality $\gamma^\beta = \gamma^0$ to switch the β-system to the 0-system through the following two equivalent hybrids.

$\mathsf{H}_{\ell,1}^\beta$: This hybrid is the same as $\mathsf{H}_{\ell-1}^\beta$ except that $\mathbf{X}_j$ in the ℓ-th ciphertext query is defined as

$$\mathbf{X}_j^\ell = \begin{cases} (\underline{0^{n_0 k},\ 0^{n_1}, 0^k,\ 0^k, 0^m}, 0^m, 1, 0) & (j=1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \underline{0^{n_1}}, \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (j>1) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^\beta, \mathbf{q}^0, \underline{d_j}, 0) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m, 0^m, \underline{d_j}, 0) & (1<j\leq s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m, 0^m, \underline{d_j}, 0) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m, 0^m, \underline{d_j}, 0) & (s+1<j) \end{cases}$$

where $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$, $\widetilde{v} \leftarrow \mathbb{Z}_p$ (if $\mathbf{y}'^{\ell} \neq \star$), $\widetilde{v} = 0$ (if $\mathbf{y}'^{\ell} = \star$) and

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{v} g(\mathbf{y}^\ell) + \sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \widetilde{\mathbf{t}}). \tag{3.1}$$

$\mathsf{H}_{\ell,2}^\beta$**:** This hybrid is the same as $\mathsf{H}_{\ell,1}^\beta$ except that $(d_1, \ldots, d_t)$ in Eq. (3.1) is defined as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{v} g(\mathbf{y}^\ell) + \sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,0} \rangle + \langle \mathbf{p}^{\ell,0}, \mathbf{q}^0 \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \widetilde{\mathbf{t}}).$$

We prove that $\mathsf{H}_{\ell-1}^\beta \approx_c \mathsf{H}_{\ell,1}^\beta = \mathsf{H}_{\ell,2}^\beta \approx_c \mathsf{H}_\ell^\beta$. We can see that $\mathsf{H}_{\ell,1}^\beta = \mathsf{H}_{\ell,2}^\beta$ by considering the two cases. If $g(\mathbf{y}'^{\ell}) = 0$ (that is, $g(\mathbf{y}^\ell) = 0$ or $\widetilde{v} = 0$), we have $\sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle = \sum_{j \in [N^{(\ell)}]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,0} \rangle + \langle \mathbf{p}^{\ell,0}, \mathbf{q}^0 \rangle$ due to the admissibility of the adversary. Otherwise, the term $\widetilde{v} g(\mathbf{y}^\ell)$ is uniformly distributed in $\mathbb{Z}_p$ and works as a one-time pad.

Proving $\mathsf{H}_{\ell-1}^\beta \approx_c \mathsf{H}_{\ell,1}^\beta$ and $\mathsf{H}_{\ell,2}^\beta \approx_c \mathsf{H}_\ell^\beta$ are similar, and we prove only the former. To this end, we introduce further intermediate hybrids $\widehat{\mathsf{H}}_{\ell,\nu,1}^\beta, \ldots, \widehat{\mathsf{H}}_{\ell,\nu,5}^\beta$ for $\nu \in [N^{(\ell)}]$ and show that $\mathsf{H}_{\ell-1}^\beta \approx_c \widehat{\mathsf{H}}_{\ell,1,1}^\beta \approx_c \cdots \approx_c \widehat{\mathsf{H}}_{\ell,1,5}^\beta \approx_c \widehat{\mathsf{H}}_{\ell,2,1}^\beta \approx_c \cdots \approx_c \widehat{\mathsf{H}}_{\ell,N^{(\ell)},5}^\beta = \mathsf{H}_{\ell,1}^\beta$. Intuitively, what we are doing in these steps is to move the information of $\mathbf{z}_\nu^{\ell,\beta}$ from $\mathbf{X}_\nu^\ell$ to $\{\mathbf{Y}_j\}_{j \in [t]}$ step by step for $\nu \in [N^\ell]$. Each hybrid is defined as follows.

$\widehat{\mathsf{H}}_{\ell,\nu,1}^\beta$**:** This hybrid is the same as $\mathsf{H}_{\ell-1}^\beta$ except that $\mathbf{X}_j$ in the ℓ-th ciphertext query is defined as

$$\mathbf{X}_j^\ell = \begin{cases} (\quad 0^{n_0 k}, \;\; 0^{n_1}, 0^k, \; 0^k, 0^m, 0^m, 1, 0) & (j = 1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, 0^{n_1}, \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (1 < j < \nu) \\ (\underline{\quad 0^{n_0 k}, \;\; 0^{n_1}, 0^k}, \; 0^k, 0^m, 0^m, 0, \underline{1}) & (1 < j = \nu) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}_j^\beta, \; \mathbf{w}_j, \; 0^k, 0^m, 0^m, 0, 0) & (\nu < j) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \;\; 0^k, \mathbf{q}^\beta, \; \mathbf{q}^0, \; \underline{d_j, d_j'}) & (j = 1) \\ (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m, 0^m, \underline{d_j, d_j'}) & (1 < j \leq s) \\ (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), 0^{n_1}, \quad 0^k, \; \mathbf{s}, \;\; 0^m, 0^m, \underline{d_j, d_j'}) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j \mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^m, \underline{d_j, d_j'}) & (s+1 < j) \end{cases}$$

where $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$, $\widetilde{v} \leftarrow \mathbb{Z}_p$ (if $\mathbf{y}'^{\ell} \neq \star$), $\widetilde{v} = 0$ (if $\mathbf{y}'^{\ell} = \star$) and

$$(d_1, \ldots, d_t) = \begin{cases} \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), (\langle \mathbf{s}, \mathbf{v}^\ell \rangle, \mathbf{z}_1^{\ell,\beta}), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \mathbf{T}\mathbf{u}_1^\ell) & (\nu = 1) \\ \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{v} g(\mathbf{y}^\ell) + \sum_{j \in [\nu-1]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \widetilde{\mathbf{t}}) & (\nu > 1) \end{cases}$$

$$(d_1', \ldots, d_t') = \begin{cases} \mathbf{0} & (\nu = 1) \\ \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_\nu^\ell), (0, \mathbf{z}_\nu^{\ell,\beta}), \langle \mathbf{w}_\nu^\ell, \mathbf{r} \rangle; \mathbf{T}\mathbf{u}_\nu^\ell) & (\nu > 1) \end{cases}$$

$\widehat{\mathsf{H}}_{\ell,\nu,2}^\beta$**:** This hybrid is the same as $\widehat{\mathsf{H}}_{\ell,\nu,1}^\beta$ except that d_i, d_i' for $i \in [t]$ is defined as

$$(d_1, \ldots, d_t) = \begin{cases} \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), (\underline{\widetilde{v}}, \mathbf{z}_1^{\ell,\beta}), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \langle \mathbf{w}_1^\ell, \mathbf{r} \rangle; \underline{\widetilde{\mathbf{t}}}) & (\nu = 1) \\ \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}_1^\ell), \widetilde{v} g(\mathbf{y}^\ell) + \sum_{j \in [\nu-1]} \langle h(\mathbf{x}_j^\ell), \mathbf{z}_j^{\ell,\beta} \rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta \rangle + \underline{\widetilde{r}_1}; \widetilde{\mathbf{t}}) & (\nu > 1) \end{cases}$$

$$(d_1', \ldots, d_t') = \mathsf{pgb}^+(\phi, (\mathbf{y}^\ell, \mathbf{x}_\nu^\ell), (0, \mathbf{z}_\nu^{\ell,\beta}), \underline{\widetilde{r}_\nu; \widetilde{\mathbf{t}}'}) \quad (\nu > 1)$$

where $\widetilde{\mathbf{t}}, \widetilde{\mathbf{t}}' \leftarrow \mathbb{Z}_p^{t-1}$, $\widetilde{r}_1, \leftarrow \mathbb{Z}_p$ and $\widetilde{r}_\nu = -\widetilde{r}_1 - \sum_{i \in [N^\ell] \setminus \{1,\nu\}} \langle \mathbf{w}_i^\ell, \mathbf{r} \rangle$.

$\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,3}$**:** This hybrid is the same as $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ except that d_i, d'_i for $i \in [t]$ is defined as

$$(d_1, \ldots, d_t) = \underline{\mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), \widetilde{v}g(\mathbf{y}^\ell) + \langle h(\mathbf{x}^\ell_1), \mathbf{z}^{\ell,\beta}_1\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle + \langle \mathbf{w}^\ell_1, \mathbf{r}\rangle; \widetilde{\mathbf{t}})} \quad (\nu = 1)$$
$$(d'_1, \ldots, d'_t) = \underline{\mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_\nu), \langle h(\mathbf{x}^\ell_\nu), \mathbf{z}^{\ell,\beta}_\nu\rangle + \widetilde{r}_\nu; \widetilde{\mathbf{t}}')} \quad (\nu > 1)$$

$\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,4}$**:** For $\nu = 1$, this hybrid is the same as $\widehat{\mathsf{H}}^{\beta}_{\ell,1,3}$. Otherwise, this hybrid is the same as $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,3}$ except that d_i, d'_i for $i \in [t]$ is defined as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), \widetilde{v}g(\mathbf{y}^\ell) + \sum_{j\in[\nu]} \langle h(\mathbf{x}^\ell_j), \mathbf{z}^{\ell,\beta}_j\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle + \widetilde{r}_1; \widetilde{\mathbf{t}}) \quad (\nu > 1)$$
$$(d'_1, \ldots, d'_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_\nu), \cancel{\langle h(\mathbf{x}^\ell_\nu), \mathbf{z}^{\ell,\beta}_\nu\rangle} + \widetilde{r}_\nu; \widetilde{\mathbf{t}}') \quad (\nu > 1)$$

$\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,5}$**:** For $\nu = 1$, this hybrid is the same as $\widehat{\mathsf{H}}^{\beta}_{\ell,1,4}$. Otherwise, this hybrid is the same as $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,4}$ except that $\mathbf{X}_j$ in the ℓ-th ciphertext query is defined as

$$\mathbf{X}^\ell_j = \begin{cases} (\quad 0^{n_0 k}, \;\; 0^{n_1}, 0^k, \;\; 0^k, 0^m, 0^m, 1, 0) & (j = 1) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, 0^{n_1}, \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (1 < j \leq \nu) \\ (\boldsymbol{\chi}_j \otimes \mathbf{u}_j, \mathbf{z}^\beta_j, \;\; \mathbf{w}_j, 0^k, 0^m, 0^m, 0, 0) & (\nu < j) \end{cases}$$

and $\mathbf{Y}_j$ for all queries are defined as

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \;\; 0^k, \mathbf{q}^\beta, \; \mathbf{q}^0, \; \underline{d_j, 0}) & (j = 1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m, 0^m, \underline{d_j, 0}) & (1 < j \leq s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, \; \mathbf{s}, \;\; 0^m, 0^m, \underline{d_j, 0}) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, 0^m, \underline{d_j, 0}) & (s+1 < j) \end{cases}$$

where $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$ and

$$(d_1, \ldots, d_t) = \mathsf{pgb}^*(\phi, (\mathbf{y}^\ell, \mathbf{x}^\ell_1), \widetilde{v}g(\mathbf{y}^\ell) + \sum_{j\in[\nu]} \langle h(\mathbf{x}^\ell_j), \mathbf{z}^{\ell,\beta}_j\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^\beta\rangle + \underline{\langle \mathbf{w}^\ell_1, \mathbf{r}\rangle}; \widetilde{\mathbf{t}})$$

Thanks to Lemmata 3.3 to 3.7, Lemma 3.2 holds. □

Lemma 3.3. *Let* $\widehat{\mathsf{H}}^{\beta}_{\ell,0,5} = \mathsf{H}^{\beta}_{\ell-1}$. *For all* $\nu \in [N^{(\ell)}]$, *we have* $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu-1,5} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ *if* iFE *is function-hiding.*

Proof. Observe that the difference of $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu-1,5}$ and $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ is described as the two cases:

$\nu = 1$**:** In this case, $\mathbf{X}^\ell_1$ in the ℓ-th ciphertext and $\mathbf{Y}_j$ in all the secret keys in $\widehat{\mathsf{H}}^{\beta}_{\ell,0,5} = \mathsf{H}^{\beta}_{\ell-1}$ are defined as follows:

$$\mathbf{X}^\ell_1 = \quad (\boldsymbol{\chi}_1 \otimes \mathbf{u}_1, \quad \mathbf{z}^\beta_1, \quad \mathbf{w}_1, \mathbf{v}, \;\; \mathbf{p}^\beta, \; 0^{m+2})$$
$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \;\; 0^k, \mathbf{q}^\beta, \;\; 0^{m+2}) & (j = 1, \ell = 1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad \mathbf{r}, \;\; 0^k, \mathbf{q}^\beta, \; \mathbf{q}^0, 0^2) & (j = 1, \ell > 1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, 0^k, 0^m, \;\; 0^{m+2}) & (1 < j \leq s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, \quad 0^k, \mathbf{s}, \;\; 0^m, \;\; 0^{m+2}) & (j = s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, 0^k, 0^k, 0^m, \;\; 0^{m+2}) & (s+1 < j) \end{cases}$$

while the corresponding vectors in $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1}$ are defined as

$$\mathbf{X}^{\ell}_1 = \quad (\underline{\quad 0^{n_0 k}, \quad 0^{n_1}, \quad 0^k, 0^k, 0^m, 0^m,\ 1,\ 0)}$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^{\beta},\ \mathbf{q}^0,\ \underline{d_j}, 0) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m,\ 0^m, \underline{d_j}, 0) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m,\ 0^m, \underline{d_j}, 0) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m,\ 0^m, \underline{d_j}, 0) & (s+1<j) \end{cases}$$

where $(d_1, \ldots, d_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), (\langle \mathbf{s}, \mathbf{v}^{\ell}\rangle, \mathbf{z}^{\ell,\beta}_1), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta}\rangle + \langle \mathbf{w}^{\ell}_1, \mathbf{r}\rangle; \mathbf{T}\mathbf{u}^{\ell}_1)$. It is not hard to see that for all secret keys and j, $\langle \mathbf{X}^{\ell}_1, \mathbf{Y}_j\rangle$ in $\mathsf{H}^{\beta}_{\ell-1}$ and that in $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1}$ are both equal to d_j. Hence, thanks to the function-hiding security of iFE, the two hybrids are indistinguishable. Note that the second to last entry of $\mathbf{X}_j$ other than $\mathbf{X}^{\ell}_1$ is 0, and thus the change of $\mathbf{Y}_j$ does not affect the other vectors.

$\nu > 1$: In this case, $\mathbf{X}^{\ell}_{\nu}$ in the ℓ-th ciphertext and $\mathbf{Y}_j$ in all the secret keys in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu-1,5}$ are defined as follows:

$$\mathbf{X}^{\ell}_{\nu} = \quad (\boldsymbol{\chi}_{\nu} \otimes \mathbf{u}_{\nu}, \quad \mathbf{z}^{\beta}_{\nu}, \quad \mathbf{w}_{\nu}, 0^k, 0^m, 0^m,\ 0,\ 0)$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^{\beta},\ \mathbf{q}^0,\ d_j, 0) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m,\ 0^m, d_j, 0) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m,\ 0^m, d_j, 0) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m,\ 0^m, d_j, 0) & (s+1<j) \end{cases}$$

while the corresponding vectors in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ are defined as

$$\mathbf{X}^{\ell}_{\nu} = \quad (\underline{\quad 0^{n_0 k} \quad, 0^{n_1}, \quad 0^k}, 0^k, 0^m,\ 0^m, 0,\ \underline{1}\ )$$

$$\mathbf{Y}_j = \begin{cases} (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & \mathbf{r},\ 0^k, \mathbf{q}^{\beta},\ \mathbf{q}^0,\ d_j, \underline{d'_j}) & (j=1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, 0^k, 0^m,\ 0^m, d_j, \underline{d'_j}) & (1<j\le s) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), 0^{n_1}, & 0^k, \mathbf{s},\ 0^m,\ 0^m, d_j, \underline{d'_j}) & (j=s+1) \\ (\mathsf{vec}(\mathbf{L}_j\mathbf{T}), \mathbf{e}_{j-s-1}, & 0^k, 0^k, 0^m,\ 0^m, d_j, \underline{d'_j}) & (s+1<j) \end{cases}$$

where $(d'_1, \ldots, d'_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_{\nu}), (0, \mathbf{z}^{\ell,\beta}_{\nu}), \langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r}\rangle; \mathbf{T}\mathbf{u}^{\ell}_{\nu})$. It is not hard to see that for all secret keys and j, $\langle \mathbf{X}^{\ell}_{\nu}, \mathbf{Y}_j\rangle$ in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu-1,5}$ and that in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ are both equal to d'_j. Hence, thanks to the function-hiding security of iFE, the two hybrids are indistinguishable. Note that the last entry of $\mathbf{X}_j$ other than $\mathbf{X}^{\ell}_{\nu}$ is 0, and thus the change of $\mathbf{Y}_j$ does not affect the other vectors. □

Lemma 3.4. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ if the MDDH$_k$ assumption holds in $\mathbb{G}$.*

Proof. Observe that the difference of $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ and $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ is described as the two cases:

$\nu = 1$: In this case, d_j in $\mathbf{Y}_j$ in $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1}$ is generated as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), (\langle \mathbf{s}, \mathbf{v}^{\ell}\rangle, \mathbf{z}^{\ell,\beta}_1), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta}\rangle + \langle \mathbf{w}^{\ell}_1, \mathbf{r}\rangle; \mathbf{T}\mathbf{u}^{\ell}_1)$$

while the corresponding terms in $\widehat{\mathsf{H}}^{\beta}_{\ell,1,2}$ is generated as

$$(d_1, \ldots, d_t) = \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), (\underline{\widetilde{v}}, \mathbf{z}^{\ell,\beta}_1), \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta}\rangle + \langle \mathbf{w}^{\ell}_1, \mathbf{r}\rangle; \underline{\widetilde{\mathbf{t}}}).$$

Observe that $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1} = \widehat{\mathsf{H}}^{\beta}_{\ell,1,2}$ if $\mathbf{y}'^{\ell} = \star$. Hence, we focus on the case $\mathbf{y}'^{\ell} \neq \star$. Recall that pgb^{+} is efficiently computable if the random tape is given as group elements due to its linearity. Hence, $\widehat{\mathsf{H}}^{\beta}_{\ell,1,1} \approx_c \widehat{\mathsf{H}}^{\beta}_{\ell,1,2}$ is reduced to

$$[\{\mathbf{T}^{\kappa}, \mathbf{s}^{\kappa}, \mathbf{T}^{\kappa}\mathbf{u}_1, \langle \mathbf{s}^{\kappa}, \mathbf{v}^{\ell}\rangle\}_{\kappa\in[q_k]}]_2 \approx_c [\{\mathbf{T}^{\kappa}, \mathbf{s}^{\kappa}, \widetilde{\mathbf{t}}^{\kappa}, \widetilde{v}^{\kappa}\}_{\kappa\in[q_k]}]_2$$

where q_k is the number of queries to $\mathsf{QKeyGen}^{\beta}$, which are essentially what the MDDH_k assumption asserts. Thanks to the linearity of pgb^{+}, this reduction is efficient.

$\nu > 1$: In this case, d_j and d'_j in $\mathbf{Y}_j$ in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ are generated as

$$\begin{aligned}(d_1,\dots,d_t) =& \mathsf{pgb}^{*}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), \widetilde{v}g(\mathbf{y}^{\ell}) + \sum_{j\in[\nu-1]} \langle h(\mathbf{x}^{\ell}_j), \mathbf{z}^{\ell,\beta}_j\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta}\rangle + \langle \mathbf{w}^{\ell}_1, \mathbf{r}\rangle; \widetilde{\mathbf{t}})\\ (d'_1,\dots,d'_t) =& \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_{\nu}), (0, \mathbf{z}^{\ell,\beta}_{\nu}), \langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r}\rangle; \mathbf{T}\mathbf{u}^{\ell}_{\nu})\end{aligned}$$

while the corresponding term in $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ are generated as

$$\begin{aligned}(d_1,\dots,d_t) =& \mathsf{pgb}^{*}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_1), \widetilde{v}g(\mathbf{y}^{\ell}) + \sum_{j\in[\nu-1]} \langle h(\mathbf{x}^{\ell}_j), \mathbf{z}^{\ell,\beta}_j\rangle + \langle \mathbf{p}^{\ell,\beta}, \mathbf{q}^{\beta}\rangle + \underline{\widetilde{r}_1}; \widetilde{\mathbf{t}})\\ (d'_1,\dots,d'_t) =& \mathsf{pgb}^{+}(\phi, (\mathbf{y}^{\ell}, \mathbf{x}^{\ell}_{\nu}), (0, \mathbf{z}^{\ell,\beta}_{\nu}), \underline{\widetilde{r}_{\nu}; \widetilde{\mathbf{t}}'})\end{aligned}$$

The indistinguishability between $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,1}$ and $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2}$ can be shown by two steps. The first step changes the random tape in pgb^{+} from $\mathbf{T}\mathbf{u}_{\nu}$ to $\widetilde{\mathbf{t}}'$. This can be proven in the same way as the case $\nu = 1$. The second step changes $\langle \mathbf{w}^{\ell}_1, \mathbf{r}\rangle$ and $\langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r}\rangle$ to $\widetilde{r}_1$ and $\widetilde{r}_{\nu}$. In this step, we would like to prove that

$$\begin{aligned}&(\{[\mathbf{r}^{\kappa}]_2, [\langle \mathbf{w}^{\ell}_1, \mathbf{r}^{\kappa}\rangle]_2, [\langle \mathbf{w}^{\ell}_{\nu}, \mathbf{r}^{\kappa}\rangle]_2\}_{\kappa\in[q_k]}, \{\mathbf{w}^{\ell}_j\}_{j\in[N^{\ell}]\setminus\{1,\nu\}})\\ &\quad\approx_c (\{[\mathbf{r}^{\kappa}]_2, [\widetilde{r}^{(\kappa)}_1]_2, [\widetilde{r}^{(\kappa)}_{\nu}]_2\}_{\kappa\in[q_k]}, \{\mathbf{w}^{\ell}_j\}_{j\in[N^{\ell}]\setminus\{1,\nu\}})\end{aligned}$$

where $\mathbf{r}^{\kappa} \leftarrow \mathbb{Z}_p^k$, $\mathbf{w}^{\ell}_1, \dots, \mathbf{w}^{\ell}_{N^{(\ell)}} \leftarrow \mathbb{Z}_p^k$ s.t. $\sum_{j\in[N^{(\ell)}]} \mathbf{w}^{\ell}_j = \mathbf{0}$, and $\widetilde{r}^{(\kappa)}_1, \widetilde{r}^{(\kappa)}_{\nu} \leftarrow \mathbb{Z}_p$ s.t. $\widetilde{r}^{(\kappa)}_1 + \widetilde{r}^{(\kappa)}_{\nu} + \sum_{j\in[N^{(\ell)}]\setminus\{1,\nu\}} \langle \mathbf{w}^{\ell}_j, \mathbf{r}^{\kappa}\rangle = \mathbf{0}$. It is not hard to see that the following indistinguishability suffices to prove the above indistinguishability:

$$([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, [\mathbf{A}\mathbf{m}_2]_2, \mathbf{m}_3, \dots, \mathbf{m}_d) \approx_c ([\mathbf{A}]_2, [\mathbf{s}_1]_2, [\mathbf{s}_2]_2, \mathbf{m}_3, \dots, \mathbf{m}_d)$$

where $d > 1, n$ are any natural numbers, $\mathbf{A} \leftarrow \mathbb{Z}_p^{n\times k}$, $\mathbf{m}_1, \dots, \mathbf{m}_d \leftarrow \mathbb{Z}_p^k$ s.t. $\sum_{i\in[d]} \mathbf{m}_i = \mathbf{0}$, and $\mathbf{s}_1, \mathbf{s}_2 \leftarrow \mathbb{Z}_p^n$ s.t. $\mathbf{s}_1 + \mathbf{s}_2 + \sum_{i\in\{3,d\}} \mathbf{A}\mathbf{m}_i = \mathbf{0}$. It is easy to see that they are distributed the same if $n \le k$, so we consider the case $n > k$. The above relation can be rewritten as

$$\begin{aligned}&([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, [-\mathbf{A}\mathbf{m}_1 - \mathbf{b}]_2, \mathbf{m}_3, \dots, \mathbf{m}_d)\\ &\quad\approx_c ([\mathbf{A}]_2, [\mathbf{s}_1]_2, [-\mathbf{s}_1 - \mathbf{b}]_2, \mathbf{m}_3, \dots, \mathbf{m}_d)\end{aligned}$$

where $\mathbf{b} = \sum_{i\in\{3,d\}} \mathbf{A}\mathbf{m}_i$. Hence, this is implied by the MDDH_k assumption, which assert that $([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2) \approx_c ([\mathbf{A}]_2, [\mathbf{s}_1]_2)$. Thanks to the linearity of pgb^{+} and pgb^{*}, this reduction is efficient. □

Lemma 3.5. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,2} \approx_s \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,3}$ if the partial garbling scheme is secure.*

Proof. The lemma readily follows from the security of the extension of the partial garbling scheme. □

Lemma 3.6. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^{\beta}_{\ell,\nu,3} = \widehat{\mathsf{H}}^{\beta}_{\ell,\nu,4}$.*

Proof. Recall that $\widetilde{r}_1$ and $\widetilde{r}_\nu$ are randomly distributed such that $\widetilde{r}_1 + \widetilde{r}_\nu = -\sum_{i\in[N^\ell]\setminus\{1,\nu\}}\langle \mathbf{w}_i^\ell, \mathbf{r}\rangle$. Therefore, $\widetilde{r}'_1 = \widetilde{r}_1 + \langle h(\mathbf{x}_\nu^\ell), \mathbf{z}_\nu^{\ell,\beta}\rangle$ and $\widetilde{r}'_\nu = \widetilde{r}_\nu - \langle h(\mathbf{x}_\nu^\ell), \mathbf{z}_\nu^{\ell,\beta}\rangle$ are also randomly distributed such that $\widetilde{r}'_1 + \widetilde{r}'_\nu = -\sum_{i\in[N^\ell]\setminus\{1,\nu\}}\langle \mathbf{w}_i^\ell, \mathbf{r}\rangle$. By applying this replacement, we can see that both hybrids are identical. □

Lemma 3.7. *For all $\nu \in [N^{(\ell)}]$, we have $\widehat{\mathsf{H}}^\beta_{\ell,\nu,4} \approx_c \widehat{\mathsf{H}}^\beta_{\ell,\nu,5}$ if iFE is function-hiding, the partial garbling scheme is secure, and the MDDH$_k$ assumption holds in $\mathbb{G}$.*

Proof. The proof of this lemma is similar to that of $\widehat{\mathsf{H}}^\beta_{\ell,\nu-1,5} \approx_c \widehat{\mathsf{H}}^\beta_{\ell,\nu,3}$. □

4 Attribute-Based MIFE for Attribute-Weighted Sums

In this section, we present our AB-MIFE for AWS in two steps as discussed in Section 1. In Appendix B, we show how it can be captured in the context of MPFE.

Definition 4.1 (Multi-Input Functional Encryption). Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X}^n \to \mathcal{Z}$.[10] An MIFE scheme for $\mathcal{F}$ consists of four algorithms.

$\mathsf{Setup}(1^\lambda, 1^n)$**:** It takes a security parameter 1^λ and a number 1^n of slots, and outputs a public parameter PP, encryption keys $\{\mathsf{EK}_i\}_{i\in[n]}$, a master secret key MSK. The other algorithms implicitly take PP.

$\mathsf{Enc}(\mathsf{EK}_i, x_i)$**:** It takes EK_i and $x_i \in \mathcal{X}$ and outputs a ciphertext CT_i.

$\mathsf{KeyGen}(\mathsf{MSK}, f)$**:** It takes MSK and $f \in \mathcal{F}$, and outputs a secret key SK.

$\mathsf{Dec}(\mathsf{CT}_1, \dots, \mathsf{CT}_n, \mathsf{SK})$**:** It takes $\mathsf{CT}_1, \dots, \mathsf{CT}_n$ and SK, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. An MIFE scheme is correct if it satisfies the following condition. For all $\lambda, n \in \mathbb{N}$, $(x_1, \dots, x_n) \in \mathcal{X}^n$, $f \in \mathcal{F}$, we have

$$\Pr\left[d = f(x_1,\dots,x_n) : \begin{array}{l}(\mathsf{PP}, \{\mathsf{EK}_i\}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)\\ \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{EK}_i, x_i) \ \text{ for } i \in [n]\\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f)\\ d = \mathsf{Dec}(\mathsf{CT}_1, \dots, \mathsf{CT}_n, \mathsf{SK})\end{array}\right] = 1.$$

Security. We consider the case where each $x_i \in \mathcal{X}$ consists of a public part $x_{i,\mathsf{pub}}$ and a private part $x_{i,\mathsf{priv}}$, i.e., $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}})$. An MIFE scheme is selectively partially-hiding if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\beta = \beta' : \begin{array}{l}\beta \leftarrow \{0,1\}\\ (\mathsf{PP}, \{\mathsf{EK}_i\}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)\\ \beta' \leftarrow \mathcal{A}^{\mathsf{QCor}(),\mathsf{QEnc}^\beta(),\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{PP})\end{array}\right] \le \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{QCor}(i)$ outputs EK_i, and $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ outputs $\mathsf{Enc}(\mathsf{EK}_i, x_i^\beta)$. Let $q_{c,i}$ be the numbers of queries of the forms of $\mathsf{QEnc}^\beta(i, *, *)$. Let $\mathcal{HS}$ be the set of parties on which the adversary has not queried QCor at the end of the game, and $\mathcal{CS} = [n]\setminus\mathcal{HS}$. Then, the admissible adversary's queries must satisfy the following conditions.

- For $i \in \mathcal{CS}$, the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ must satisfy $x_i^0 = x_i^1$.
- For $i \in \mathcal{HS}$, the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ must satisfy $x^0_{i,\mathsf{pub}} = x^1_{i,\mathsf{pub}}$.

[10] In general, the domain of each slot can be different, i.e., f can be defined as $f : \mathcal{X}_1 \cdots \times \mathcal{X}_n \to \mathcal{Z}$. In this paper, however, we only handle the case where $\mathcal{X}_i = \mathcal{X}$ for all $i \in [n]$.

– $f(x_1^0, \ldots, x_n^0) = f(x_1^1, \ldots, x_n^1)$ for all sequences $(x_1^0, \ldots, x_n^0, x_1^1, \ldots, x_n^1, f)$ that satisfy the two conditions:
  - For all $i \in [n]$, $[\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ is queried and $i \in \mathcal{HS}]$ or $[x_i^0 = x_i^1 \in \mathcal{X}_i$ and $i \in \mathcal{CS}]$.
  - $\mathsf{KeyGen}(\mathsf{MSK}, f)$ is queried.
– The adversary must make all queries to QCor and QEnc in one shot. That is, first it outputs $(\mathcal{CS}, \{i, x_i^0, x_i^1\})$ and obtains the response: $(\{\mathsf{EK}_i\}_{i\in\mathcal{CS}}, \{\mathsf{Enc}(\mathsf{EK}_i, x_i^\beta)\})$. Only after the one-shot query, the adversary can query KeyGen adaptively.

We formally define attribute-based MIFE scheme for attribute-weighted sums and its security.

Definition 4.2 (AB-MIFE for AWS). Attribute-based MIFE for Attribute-Weighted Sums (AB-MIFE for AWS) is a class of MIFE (Definition 4.1) that supports the following functionality. Let $\mathbb{G}$ be bilinear groups. Let $\mathcal{X} = (\mathbb{Z}_p^{n_0'} \cup \{\star\}) \times \bigcup_{i\in\mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i$ be a message space. Let $\mathcal{F} = (\mathcal{F}^{\mathsf{ABP}}_{n_0',1} \times \mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^n$ be a family of functions, where $((g_1, h_1), \ldots, (g_n, h_n)) \in \mathcal{F}$ represents the function $f : \mathcal{X}^n \to G_T$ defined as

$$f((\mathbf{y}_1, \{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j\in[N_1]}), \ldots, (\mathbf{y}_n, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j\in[N_n]})) = \begin{cases} [\sum_{i\in[n]} \sum_{j\in[N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle]_T & (g_i(\mathbf{y}_i) = 0 \text{ for all } i \in [n]) \\ \bot & (\text{otherwise}) \end{cases}$$

where $\mathbf{y}_i, \mathbf{x}_{i,j}$ are public inputs while $\mathbf{z}_{i,j}$ is a private input, and we always have $g_i(\star) = 0$.

Definition 4.3 (Security of AB-MIFE for AWS). We say that an AB-FE scheme for AWS satisfies security against *legitimate* keys if the scheme is secure against adversaries that follows the condition defined below in addition to the conditions defined in Definition 4.1. Let $(\mathcal{CS}, \{i, x_i^{\ell,0}, x_i^{\ell,1}\}_{i\in[n],\ell\in[q_{c,i}]}, \{f^\eta\}_{\eta\in[q_k]})$ be the query of the adversary, where q_k is the number of queries to KeyGen, $x_i^{\ell,\beta} = (\mathbf{y}_i^\ell, \{\mathbf{x}_{i,j}^\ell, \mathbf{z}_{i,j}^{\ell,\beta}\}_{j\in[N_i^\ell]})$ and $f^\eta = \{g_i^\eta, h_i^\eta\}_{i\in[n]}$. We say that f^η is legitimate if for all $i \in \mathcal{HS}$, there exists $\ell_i' \in [q_{c,i}]$ such that $g_i^\eta(\mathbf{y}_i^{\ell_i'}) = 0$. In security against legitimate keys, f^η must be legitimate for all $\eta \in [q_k]$. In contrast, we say that an AB-FE scheme for AWS satisfies security against *any* keys if the scheme is secure against adversaries that follows just the condition defined in Definition 4.1.

4.1 Construction

Let $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ be an FE scheme for AB-FE for AWSw/IP. Then our AB-MIFE scheme for AWS is given in Figure 4.

Correctness and Security. Due to the correctness of aFE, we have

$$d_i = \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + r_i$$

Hence $d = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$ since $\sum_{i\in[n]} r_i = 0$.

The proposed scheme is secure against legitimate keys as stated by the following theorem.

Theorem 4.1. *If* aFE *is partially function-hiding, then the proposed AB-MIFE scheme for AWS satisfies security against legitimate keys as per Definition 4.3.*

Proof. We prove the theorem via two hybrids H_1^β and H_2^β. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta = \mathsf{H}_2^\beta$, where H_s^β for $\beta \in \{0, 1\}$ is the original security game. Recall that in H_s^β the challenger replies

$$\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]}, [1]_1)) \quad \text{and} \quad \{\mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i]_2))\}_{i\in[n]}$$

for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, respectively, where

$$x_i^\beta = (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]}) \quad \text{and} \quad f = \{g_i, h_i\}_{i\in[n]}.$$

$\mathsf{Setup}(1^\lambda, 1^n)$**:** It runs $\mathsf{aPP}_i, \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$ and outputs

$$\mathsf{PP} = \{\mathsf{aPP}_i\}_{i\in[n]},\ \mathsf{EK}_i = \mathsf{aMSK}_i \ \text{ for } i \in [n],\ \mathsf{MSK} = \{\mathsf{EK}_i\}_{i\in[n]}.$$

$\mathsf{Enc}(\mathsf{EK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}))$**:** It outputs

$$\mathsf{CT}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}, [1]_1)).$$

$\mathsf{KeyGen}(\mathsf{MSK}, \{g_i, h_i\}_{i\in[n]})$**:** It samples $r_1, \ldots, r_{n-1} \leftarrow \mathbb{Z}_p$, sets $r_n = -\sum_{i\in[n-1]} r_i$, and outputs $\mathsf{SK} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where

$$\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i]_2)).$$

$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})$**:** It parse $\mathsf{CT}_i, \mathsf{SK}$ as $\mathsf{aCT}_i, \{\mathsf{aSK}_i\}_{i\in[n]}$, respectively. If there exists i such that $g_i(\mathbf{y}_i) \neq 0$, it outputs $\perp$. Otherwise, it computes $[d_i]_T = \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i)$ for $i \in [n]$, and outputs $[d]_T = \sum_{i\in[n]} [d_i]_T$.

Fig 4. Attribute-Based MIFE for AWS

The hybrid H_1^β is the same as H_s^β except that for all $i \in \mathcal{HS}$, the challenger replies

$$\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \underline{\mathbf{z}^0_{i,j}}\}_{j\in[N_i]}, [1]_1)) \quad \text{and}$$
$$\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i + \underline{\sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,\beta}_{i,j}\rangle - \sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,0}_{i,j}\rangle}]_2))\}_{i\in[n]}$$

for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, respectively, where $\{\mathbf{x}^{\ell_i}_{i,j}, \mathbf{z}^{\ell_i,0}_{i,j}, \mathbf{z}^{\ell_i,1}_{i,j}\}_{j\in[N_i^{\ell_i}]}$ are the components of the ℓ_i-th challenge message, and $\ell_i = \min\{\ell' \in [q_{c,i}] \mid g_i(\mathbf{y}_i^{\ell'}) = 0\}$ for $i \in \mathcal{HS}$. Since all secret keys are legitimate, such ℓ_i always exists for each key query.

The hybrid H_2^β is the same as H_1^β except that for all $i \in \mathcal{HS}$, the challenger replies

$$\mathsf{aEnc}(\mathsf{aMSK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}^0_{i,j}\}_{j\in[N_i]}, [1]_1)) \quad \text{and}$$
$$\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, (g_i, h_i, [r_i + \sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,\beta}_{i,j}\rangle - \cancel{\sum_{j\in[N_i^{\ell_i}]} \langle h_i(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,0}_{i,j}\rangle}]_2))\}_{i\in[n]}$$

for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, respectively. Note that the advantage of the adversary is 0 in H_2^β since it does not obtain the information of β. Hence the theorem follows from Lemmata 4.1 and 4.2. □

Lemma 4.1. *We have* $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta$ *if* aFE *is partially function-hiding.*

Proof. Let $q_{c.i}$ be the number of the ciphertext queries for slot i, and q_k be the number of the secret key queries. For $\mu \in [q_{c,i}]$ and $\nu \in [q_k]$, let aCT_i^μ be the μ-th challenge ciphertext for slot i, and aSK_i^ν be the i-th element of the ν-th secret key. For $j \in \{s, 1\}$, let $\delta_{j,i}^{\mu,\nu} = \mathsf{aDec}(\mathsf{aCT}_i^\mu, \mathsf{aSK}_i^\nu)$ be the decryption value in H_j^β. Then, what we need to prove is $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu}$ for all $i \in \mathcal{HS}, \mu \in [q_{c,i}], \nu \in [q_k]$.

This can be proven by the three cases.

- If $g_i^\nu(\mathbf{y}_i^\mu) \neq 0$, we have $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu} = \perp$ for all $i \in \mathcal{HS}, \mu \in [q_{c,i}], \nu \in [q_k]$.
- If $\mu = \ell_i$, we have $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu} = \sum_{j\in[N_i^{\ell_i}]} \langle h_i^\nu(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,\beta}_{i,j}\rangle$ for all $i \in \mathcal{HS}, \nu \in [q_k]$.
- If $\mu > \ell_i$ and $g_i^\nu(\mathbf{y}_i^\mu) = 0$, we have $\delta_{s,i}^{\mu,\nu} = \sum_{j\in[N_i^{\ell_i}]} \langle h_i^\nu(\mathbf{x}^{\mu}_{i,j}), \mathbf{z}^{\mu,\beta}_{i,j}\rangle$ and

$$\delta_{1,i}^{\mu,\nu} = \sum_{j\in[N_i^{\ell_i}]} (\langle h_i^\nu(\mathbf{x}^{\mu}_{i,j}), \mathbf{z}^{\mu,0}_{i,j}\rangle + \langle h_i^\nu(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,\beta}_{i,j}\rangle - \langle h_i^\nu(\mathbf{x}^{\ell_i}_{i,j}), \mathbf{z}^{\ell_i,0}_{i,j}\rangle)$$

for all $i \in \mathcal{HS}, \nu \in [q_k]$. We can prove $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu}$ as follows. Due to the admissibility of the adversary, we have

$$\sum_{k\in\mathcal{HS}} \sum_{j\in[N_k^{\ell_k}]} \langle h_k^{\nu}(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,0}\rangle = \sum_{k\in\mathcal{HS}} \sum_{j\in[N_k^{\ell_k}]} \langle h_k^{\nu}(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,1}\rangle \tag{4.1}$$

$$\begin{aligned} &\sum_{j\in[N_i^{\ell_i}]} \langle h_i^{\nu}(\mathbf{x}_{i,j}^{\mu}), \mathbf{z}_{i,j}^{\mu,0}\rangle + \sum_{k\in\mathcal{HS}\setminus\{i\}} \sum_{j\in[N_k^{\ell_k}]} \langle h_k^{\nu}(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,0}\rangle \\ &= \sum_{j\in[N_i^{\ell_i}]} \langle h_i^{\nu}(\mathbf{x}_{i,j}^{\mu}), \mathbf{z}_{i,j}^{\mu,1}\rangle + \sum_{k\in\mathcal{HS}\setminus\{i\}} \sum_{j\in[N_k^{\ell_k}]} \langle h_k^{\nu}(\mathbf{x}_{k,j}^{\ell_k}), \mathbf{z}_{k,j}^{\ell_k,1}\rangle \end{aligned} \tag{4.2}$$

We can readily obtain $\delta_{s,i}^{\mu,\nu} = \delta_{1,i}^{\mu,\nu}$ by subtracting Eq. (4.1) from Eq. (4.2) in the third case.

□

Lemma 4.2. $\mathsf{H}_1^{\beta} = \mathsf{H}_2^{\beta}$.

Proof. From Eq. (4.1), the following distributions are identical:

$$\left\{ (r_1,\dots,r_n) \;:\; r_1,\dots,r_{n-1} \leftarrow \mathbb{Z}_p,\ r_n = -\sum_{i\in[n-1]} r_i \right\} \quad \text{and}$$

$$\left\{ (r_1,\dots,r_n) \;:\; \begin{array}{l} r'_1,\dots,r'_{n-1} \leftarrow \mathbb{Z}_p,\ r'_n = -\sum_{i\in[n-1]} r'_i \\ r_i = \begin{cases} r'_i + \sum_{j\in[N_i^{\ell_i}]} (\langle h_i^{\nu}(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,\beta}\rangle - \langle h_i^{\nu}(\mathbf{x}_{i,j}^{\ell_i}), \mathbf{z}_{i,j}^{\ell_i,0}\rangle) & (i \in \mathcal{HS}) \\ r'_i & (i \in \mathcal{CS}) \end{cases} \end{array} \right\}$$

Hence H_1^{β} and H_2^{β} are identically distributed. □

4.2 Security against Any Keys in AB-MIFE for AWS

In this section, we present how to convert an AB-MIFE scheme for AWS with security against legitimate keys to one with security against any keys. In the conversion, we use a ciphertext-policy ABE (CP-ABE) scheme for ABP and a (n-out-of-n) secret sharing scheme. A CP-ABE scheme for ABP with wildcards is an ABE scheme (Definition 2.8) that supports predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ where $\mathcal{X} = \mathcal{F}_{n_0,1}^{\mathsf{ABP}}$, $\mathcal{Y} = \mathbb{Z}_p^{n_0} \cup \{\star\}$, and for $g \in \mathcal{X}, \mathbf{y} \in \mathcal{Y}$, P is defined as

$$\mathsf{P}(g,\mathbf{y}) = \begin{cases} 1 & g(\mathbf{y}) = 0 \\ 0 & g(\mathbf{y}) \neq 0 \end{cases}$$

A CP-ABE scheme for ABP with wildcards is easily obtained from the CP-ABE scheme for ABP in [LL20b] just by setting the master secret key as the secret key for the wildcard.

Construction. Let $\mathsf{wmFE} = (\mathsf{wmSetup}, \mathsf{wmEnc}, \mathsf{wmKeyGen}, \mathsf{wmDec})$ be an AB-MIFE scheme for AWS with security against legitimate keys, $\mathsf{ABE} = (\mathsf{abSetup}, \mathsf{abEnc}, \mathsf{abKeyGen}, \mathsf{abDec})$ be an CP-ABE scheme for ABP, and $(\mathsf{Share}, \mathsf{Rec})$ be a secret sharing scheme. Then, an AB-MIFE scheme for AWS can be constructed as in Figure 5.

Correctness and Security. Due to the correctness of ABE, $\sigma'_1,\dots,\sigma'_n$ are valid shares of wmSK for $\{g_i, h_i\}_{i\in[n]}$. Thus, thanks to the correctness of wmFE, we have $d = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$.

We argue security via the following theorem.

$\mathsf{Setup}(1^\lambda, 1^n)$: It runs $\mathsf{wmPP}, \{\mathsf{wmEK}_i\}_{i\in[n]}, \mathsf{wmMSK} \leftarrow \mathsf{wmSetup}(1^\lambda)$ and $\mathsf{abPK}_i, \mathsf{abMSK}_i \leftarrow \mathsf{abSetup}(1^\lambda)$ for $i \in [n]$. It outputs $\mathsf{PP}, \{\mathsf{EK}_i\}_{i\in[n]}, \mathsf{MSK}$ as follows:

$$\mathsf{PP} = (\mathsf{wmPP}, \{\mathsf{abPK}_i\}_{i\in[n]}), \quad \mathsf{EK}_i = (\mathsf{wmEK}_i, \mathsf{abMSK}_i), \quad \mathsf{MSK} = \mathsf{wmMSK}$$

$\mathsf{Enc}(\mathsf{EK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}))$: It outputs $\mathsf{CT}_i = (\mathsf{wmCT}_i, \mathsf{abSK}_i)$ where

$$\mathsf{wmCT}_i \leftarrow \mathsf{wmEnc}(\mathsf{wmEK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]})), \quad \mathsf{abSK}_i \leftarrow \mathsf{abKeyGen}(\mathsf{abMSK}_i, \mathbf{y}_i)$$

$\mathsf{KeyGen}(\mathsf{MSK}, \{g_i, h_i\}_{i\in[n]})$: It outputs SK as follows:

$$\mathsf{wmSK} \leftarrow \mathsf{wmKeyGen}(\mathsf{wmMSK}_i, \{g_i, h_i\}_{i\in[n]}), \quad (\sigma_1, \dots, \sigma_n) \leftarrow \mathsf{Share}(\mathsf{wmSK}, n)$$
$$\mathsf{abCT}_i \leftarrow \mathsf{abEnc}(g_i, \sigma_i) \ \text{ for } i \in [n], \quad \mathsf{SK} = \{\mathsf{abCT}_i\}_{i\in[n]}$$

$\mathsf{Dec}(\mathsf{CT}_1, \dots, \mathsf{CT}_n, \mathsf{SK})$: It parse $\mathsf{CT}_i, \mathsf{SK}$ as $(\mathsf{wmCT}_i, \mathsf{abSK}_i), \{\mathsf{abCT}_i\}_{i\in[n]}$, respectively. If there exists i such that $g_i(\mathbf{y}_i) \neq 0$, it outputs $\perp$. Otherwise, it outputs $[d]_T$ as follows:

$$\sigma'_i = \mathsf{abDec}(\mathsf{abCT}_i, \mathsf{abSK}_i) \ \text{ for } i \in [n], \quad \mathsf{wmSK}' = \mathsf{Rec}(\sigma'_1, \dots, \sigma'_n)$$
$$[d]_T = \mathsf{wmDec}(\mathsf{wmCT}_1, \dots, \mathsf{wmCT}_n, \mathsf{wmSK}')$$

Fig 5. Attribute-Based MIFE for AWS with Standard Security

Theorem 4.2. *If* wmFE *has security against legitimate keys,* ABE *is selectively secure, and the secret sharing scheme is secure, then the proposed scheme satisfies security against any keys, i.e., selectively partially-hiding security in Definition 4.1.*

Proof. We prove the theorem via three hybrids $\mathsf{H}_1^\beta, \mathsf{H}_2^\beta, \mathsf{H}_3^\beta$. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta = \mathsf{H}_2^\beta \approx_c \mathsf{H}_3^\beta$, where H_s^β for $\beta \in \{0,1\}$ is the original security game. Let us call a secret key with which all the combinations of challenge ciphertexts decrypt to $\perp$ a illegitimate key. For each illegitimate key for $\{g_i, h_i\}_{i\in[n]}$, there exists $i' \in \mathcal{HS}$ such that $g_{i'}(\mathbf{y}_{i'}^\ell) \neq 0$ for all $\ell \in [q_{c,i'}]$.

In H_1^β, we change the replies to the illegitimate-secret-key queries. Specifically, $\mathsf{abCT}_{i'}$ in SK is generated as $\mathsf{abCT}_{i'} \leftarrow \mathsf{abEnc}(g_{i'}, 0^m)$ instead of $\mathsf{abCT}_{i'} \leftarrow \mathsf{abEnc}(g_{i'}, \sigma_{i'})$, where m is the bit-length of a share. We can easily observe that $\mathsf{H}_s \approx_c \mathsf{H}_1$ due to the security of the CP-ABE scheme for ABP.

In H_2^β, we change the replies to the illegitimate-secret-key queries. Specifically, $\sigma_1, \dots, \sigma_n$ is generated as $\sigma_i \leftarrow \{0,1\}^m$ for $i \in [n]$ instead of being generated by the sharing algorithm. $\mathsf{H}_1 = \mathsf{H}_2$ directly follows from the security of the secret sharing scheme.

In H_3^β, we change the challenge ciphertexts. Instead of replying $\mathsf{Enc}(\mathsf{EK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]}))$ to ciphertext queries, the challenger replies $\mathsf{Enc}(\mathsf{EK}_i, (\mathbf{y}_i, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^0\}_{j\in[N_i]}))$ for all the queries. $\mathsf{H}_2 \approx_c \mathsf{H}_3$ directly follows from the security of wmFE. Note that the advantage of the adversary is 0 in H_3^β since it does not obtain the information of β. □

5 Multi-Client FE for Attribute-Weighted Sums

We define multi-client functional encryption, which basically follows the definition in [ABG19]. The essential difference from the definition in [ABG19] is that we add the definition of selective security.

Definition 5.1 (Multi-Client Functional Encryption). Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X}^n \to \mathcal{Z}$. Let $\mathcal{L}$ be a label space. An MCFE scheme for $\mathcal{F}$ and $\mathcal{L}$ consists of four algorithms.

$\mathsf{Setup}(1^\lambda, 1^n)$: It takes a security parameter 1^λ and a number 1^n of slots, and outputs a public parameter PP, encryption keys $\{\mathsf{EK}_i\}_{i\in[n]}$, a master secret key MSK. The other algorithms implicitly take PP.

$\mathsf{Enc}(\mathsf{EK}_i, x_i, L)$**:** It takes EK_i, an index $i \in [n]$, $x_i \in \mathcal{X}$, and a label L and outputs a ciphertext CT_i.
$\mathsf{KeyGen}(\mathsf{MSK}, f)$**:** It takes MSK and $f \in \mathcal{F}$, and outputs a secret key SK.
$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})$**:** It takes $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ and SK, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. An MCFE scheme is correct if it satisfies the following condition. For all $\lambda, n \in \mathbb{N}$, $(x_1, \ldots, x_n) \in \mathcal{X}^n$, $f \in \mathcal{F}$, $L \in \mathcal{L}$, we have

$$\Pr\left[d = f(x_1, \ldots, x_n) : \begin{array}{l} (\mathsf{PP}, \{\mathsf{EK}_i\}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{EK}_i, x_i, L) \ \text{ for } i \in [n] \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \\ d = \mathsf{Dec}(\mathsf{CT}_1, \ldots, , \mathsf{CT}_n, \mathsf{SK}) \end{array}\right] = 1.$$

Security. We consider the case where each $x_i \in \mathcal{X}$ consists of a public part $x_{i,\mathsf{pub}}$ and a private part $x_{i,\mathsf{priv}}$, i.e., $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}})$. An MCFE scheme is xx-yy-partially-hiding (xx $\in$ {sel, sta, adt}, yy $\in$ {any, pos}) if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function negl such that for all $\lambda, n \in \mathbb{N}$, the following holds

$$\Pr\left[\beta = \beta' : \begin{array}{l} \beta \leftarrow \{0,1\} \\ (\mathsf{PP}, \{\mathsf{EK}_i\}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QCor}(), \mathsf{QEnc}^\beta(), \mathsf{KeyGen}(\mathsf{MSK}, \cdot)}(\mathsf{PP}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{QCor}(i)$ outputs EK_i, and $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ outputs $\mathsf{Enc}(\mathsf{EK}_i, x_i^\beta, L)$. Let $q_{c,i,L}$ be the numbers of queries of the forms of $\mathsf{QEnc}^\beta(i, *, *, L)$. Let $\mathcal{HS}$ be the set of parties on which the adversary has not queried QCor at the end of the game, and $\mathcal{CS} = [n] \backslash \mathcal{HS}$. Then, the admissible adversary's queries must satisfy the following conditions.

- For $i \in \mathcal{CS}$, the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ must satisfy $x_i^0 = x_i^1$.
- For $i \in \mathcal{HS}$, the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ must satisfy $x_{i,\mathsf{pub}}^0 = x_{i,\mathsf{pub}}^1$.
- $f(x_1^0, \ldots, x_n^0) = f(x_1^1, \ldots, x_n^1)$ for all sequences $(x_1^0, \ldots, x_n^0, x_1^1, \ldots, x_n^1, f, L)$ that satisfy the two conditions:
  - For all $i \in [n]$, $[\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ is queried and $i \in \mathcal{HS}]$ or $[x_i^0 = x_i^1 \in \mathcal{X}_i$ and $i \in \mathcal{CS}]$.
  - $\mathsf{KeyGen}(\mathsf{MSK}, f)$ is queried.
- When xx = sta: the adversary cannot query QCor after querying QEnc or KeyGen even once.
- When xx = sel: the adversary must make all queries to QCor and QEnc in one shot. That is, first it outputs $(\mathcal{CS}, \{i, x_i^0, x_i^1, L\})$ and obtains the response: $(\{\mathsf{EK}_i\}_{i \in \mathcal{CS}}, \{\mathsf{Enc}(\mathsf{EK}_i, x_i^\beta, L)\})$. Only after the one-shot query, the adversary can query KeyGen adaptively.
- When yy = pos: for each $L \in \mathcal{L}$, either $q_{c,i,L} > 0$ for all $i \in \mathcal{HS}$ or $q_{c,i,L} = 0$ for all $i \in \mathcal{HS}$[11].

First, we formally define MCFE for AWS.

Definition 5.2 (MCFE for Attribute-Weighted Sums). MCFE for Attribute-Weighted Sums (AWS) is a class of MCFE (Definition 5.1) that supports the following functionality. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. Let $\mathcal{X} = \bigcup_{i \in \mathbb{N}} (\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i$ be a message space. Let $\mathcal{F} = (\mathcal{F}_{n_0,n_1}^{\mathsf{ABP}})^n$ be a family of functions, where $(f_1, \ldots, f_n) \in \mathcal{F}$ represents the function $f' : \mathcal{X}^n \to G_T$ defined as

$$f'(\{\mathbf{x}_{1,j}, \mathbf{z}_{1,j}\}_{j \in [N_1]}, \ldots, \{\mathbf{x}_{n,j}, \mathbf{z}_{n,j}\}_{j \in [N_n]}) = [\sum_{i \in [n]} \sum_{j \in [N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle]_T.$$

$\mathsf{Setup}(1^\lambda, 1^n)$: It runs $\mathsf{aPP}_i, \mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ for $i \in [n]$, chooses $\mathsf{K}_{i,j} \leftarrow \mathcal{K}$ for $i, j \in [n], i < j$, and sets $\mathsf{K}_{i,j} = \mathsf{K}_{j,i}$ for $j < i$. It outputs

$$\mathsf{PP} = \{\mathsf{aPP}_i\}_{i\in[n]},\ \mathsf{EK}_i = (\mathsf{aMSK}_i, \{\mathsf{K}_{i,j}\}_{j\in[n]\setminus\{i\}}) \ \text{ for } i \in [n],\ \mathsf{MSK} = \{\mathsf{EK}_i\}_{i\in[n]}.$$

$\mathsf{Enc}(\mathsf{EK}_i, L, x_i = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]})$: It computes $\mathbf{v}_{L,i} = \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L)$ and outputs

$$\mathsf{CT}_i = \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, x_i, [(\mathbf{v}_{L,i}, 0)]_1).$$

$\mathsf{KeyGen}(\mathsf{MSK}, \{f_i\}_{i\in[n]})$: It samples $\mathbf{s} \leftarrow \mathbb{Z}_p^k$ and outputs $\mathsf{SK} = \{\mathsf{aSK}_i\}_{i\in[n]}$ where

$$\mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, f_i, [(\mathbf{s}, 0)]_2).$$

$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})$: It parse $\mathsf{CT}_i, \mathsf{SK}$ as $\mathsf{aCT}_i, \{\mathsf{aSK}_i\}_{i\in[n]}$, respectively. It computes $[d_i]_T = \mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i)$ for $i \in [n]$, and outputs $[d]_T = \sum_{i\in[n]} [d_i]_T$.

Fig 6. Multi-Client FE for AWS

5.1 Construction

Let $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ be an FE scheme for AWSw/IP. Let $\mathsf{PRF}^{\mathsf{K}} : \mathcal{L} \to \mathbb{Z}_p^k$ be a PRF with key space $\mathcal{K}$. Let k be the parameter for the MDDH_k assumption. Construction of our MCFE scheme for AWS is given in Figure 6.

Correctness and Security. Due to the correctness of aFE, we have

$$d_i = \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + \langle \mathbf{v}_{L,i}, \mathbf{s}\rangle$$

Hence $d = \sum_{i\in[n]} \sum_{j\in[N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle$ since $\sum_{i\in[n]} \langle \mathbf{v}_{L,i}, \mathbf{s}\rangle = 0$.

We argue security via the following theorem.

Theorem 5.1. *If aFE is partially function-hiding, and the MDDH_k assumption holds in $\mathbb{G}$, then the proposed MCFE scheme for AWS is sel-pos-partially-hiding as per Definition 5.1.*

Proof. We prove the theorem via a series of hybrid games H_ℓ^β for $\ell \in \{0\} \cup [q_L]$ where $q_L = |\{L \mid q_{c,i,L} > 0\}|$ for $i \in \mathcal{HS}$ is the maximum number of labels queried by the adversary. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_0^\beta \approx_c \mathsf{H}_1^\beta \approx_c \cdots \approx_c \mathsf{H}_{q_L}^\beta$, where H_s^β for $\beta \in \{0, 1\}$ is the original security game. Recall that in H_s^β the challenger replies $\mathsf{aEnc}(\mathsf{aMSK}_i, x_i^\beta, [\mathbf{p}_i]_1)$ and $\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, f_i, [\mathbf{q}_i]_2)\}_{i\in[n]}$ for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, respectively, where

$$x_i^\beta = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^\beta\}_{j\in[N_i]},\ \ \mathbf{p}_i = (\mathbf{v}_{L,i}, 0),\ \ \mathbf{q}_i = (\mathbf{s}, 0).$$

Let $\mathcal{QL} = \{L_1, \ldots, L_{q_L}\}$ be the labels that are queried by the adversary. H_0^β is the same as H_s^β except that the challenger randomly chooses $\mathbf{v}_{L,i} \in \mathbb{Z}_p^k$ for $i \in \mathcal{HS}, L \in \mathcal{QL}$ such that $\sum_{i\in\mathcal{HS}} \mathbf{v}_{L,i} + \sum_{i\in\mathcal{CS}} \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L) = \mathbf{0}$. The hybrid H_ℓ^β is the same as H_0^β except that for the queries $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, L)$ such that $L \in \{L_1, \ldots, L_\ell\}$, the challenger replies $\mathsf{aEnc}(\mathsf{aMSK}_i, \underline{x_i^0}, \mathbf{p}_i)$. Note that the advantage of the adversary is 0 in $\mathsf{H}_{q_L}^\beta$ since it does not obtain the information of β. It is not hard to see that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_0^\beta$ follows from the security of the PRF. Hence, the theorem holds from Lemma 5.1. □

Lemma 5.1. *Let $\mathsf{H}_0^\beta = \mathsf{H}_s^\beta$. For all $\ell \in [q_L]$, we have $\mathsf{H}_{\ell-1}^\beta \approx_c \mathsf{H}_\ell^\beta$.*

[11] We can covert a xx-pos-partially-hiding scheme to xx-any-partially-hiding scheme generically [ABG19].

Proof. To prove the lemma we introduce intermediate hybrids $\mathsf{H}^{\beta}_{\ell,1}, \mathsf{H}^{\beta}_{\ell,2}, \mathsf{H}^{\beta}_{\ell,3}$, which are defined as follows:

$\mathsf{H}^{\beta}_{\ell,1}$**:** This hybrid is the same as $\mathsf{H}^{\beta}_{\ell-1}$ except that the challenger replies $\mathsf{aEnc}(\mathsf{aMSK}_i, x_i^0, [\mathbf{p}_i]_1)$ and $\{\mathsf{aKeyGen}(\mathsf{aMSK}_i, f_i, [\mathbf{q}_i]_2)\}_{i\in[n]}$ for the queries $\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L_\ell)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, respectively, where

$$\mathbf{p}_i = (\underline{0^k, 1}), \quad \mathbf{q}_i = (\mathbf{s}, \underline{\langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle + f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})}) \quad \text{for } i \in \mathcal{HS}.$$

where $(x^{\kappa,\beta}_{i,L_\ell}, x^{\kappa,0}_{i,L_\ell})$ is the pair of challenge messages in the κ-th query to QEnc of the form $(i, *, *, L_\ell)$.

$\mathsf{H}^{\beta}_{\ell,2}$**:** This hybrid is the same as $\mathsf{H}^{\beta}_{\ell,1}$ except that in the replies for the queries $\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L_\ell)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, $\mathbf{p}_i$ and $\mathbf{q}_i$ is defined as

$$v_{L_\ell,i} \leftarrow \mathbb{Z}_p \ \text{ for } i \in \mathcal{HS} \text{ s.t. } \sum_{i\in\mathcal{HS}} v_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle = 0$$
$$\mathbf{p}_i = (0^k, 1), \quad \mathbf{q}_i = (\mathbf{s}, \underline{v_{L_\ell,i}} + f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})) \quad \text{for } i \in \mathcal{HS}.$$

$\mathsf{H}^{\beta}_{\ell,3}$**:** This hybrid is the same as $\mathsf{H}^{\beta}_{\ell,2}$ except that in the replies for the queries $\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L_\ell)$ and $\mathsf{KeyGen}(\mathsf{MSK}, f)$, $\mathbf{p}_i$ and $\mathbf{q}_i$ is defined as

$$v_{L_\ell,i} \leftarrow \mathbb{Z}_p \ \text{ for } i \in \mathcal{HS} \text{ s.t. } \sum_{i\in\mathcal{HS}} v_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle = 0$$
$$\mathbf{p}_i = (0^k, 1), \quad \mathbf{q}_i = (\mathbf{s}, v_{L_\ell,i} + \cancel{f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})}) \quad \text{for } i \in \mathcal{HS}.$$

Thanks to Lemmata 5.2 to 5.5, Lemma 5.1 holds. □

Lemma 5.2. *Let $\mathsf{H}^{\beta}_0 = \mathsf{H}^{\beta}_s$. For all $\ell \in [q_L]$, we have $\mathsf{H}^{\beta}_{\ell-1} \approx_c \mathsf{H}^{\beta}_{\ell,1}$ if aFE is partially function-hiding.*

Proof. Observe that for all aCT_is that the adversary obtains as a reply to the query of the form $\mathsf{QEnc}^{\beta}(i, x_i^0, x_i^1, L)$ and all aSK_is that it obtains as a reply to the query of the form $\mathsf{KeyGen}(\mathsf{MSK}, f = \{f_i\}_{i\in\mathcal{HS}})$, the output of $\mathsf{aDec}(\mathsf{aCT}_i, \mathsf{aSK}_i)$ in $\mathsf{H}^{\beta}_{\ell-1}$ and that in $\mathsf{H}^{\beta}_{\ell,1}$ are equal for all $i \in \mathcal{HS}$. Here, we use the fact that for all $i \in \mathcal{HS}$ and $\kappa \in [q_{c,i,L_\ell}]$, we have

$$f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell}) = f_i(x^{\kappa,\beta}_{i,L_\ell}) - f_i(x^{\kappa,0}_{i,L_\ell}).$$

This is basically obtained by Eq. (5.1) − Eq. (5.2):

$$\sum_{i\in\mathcal{HS}} f_i(x^{1,\beta}_{i,L_\ell}) = \sum_{i\in\mathcal{HS}} f_i(x^{1,0}_{i,L_\ell}) \tag{5.1}$$
$$f_{i'}(x^{\kappa,\beta}_{i',L_\ell}) + \sum_{i\in\mathcal{HS}\setminus\{i'\}} f_i(x^{1,\beta}_{i,L_\ell}) = f_{i'}(x^{\kappa,0}_{i',L_\ell}) + \sum_{i\in\mathcal{HS}\setminus\{i'\}} f_i(x^{1,0}_{i,L_\ell}) \tag{5.2}$$

which follows from the query condition in Definition 5.1. Hence, thanks to the partially function-hiding security of aFE, these hybrids are indistinguishable. □

Lemma 5.3. *For all $\ell \in [q_L]$, we have $\mathsf{H}^{\beta}_{\ell,1} \approx_c \mathsf{H}^{\beta}_{\ell,2}$ if the MDDH$_k$ holds in $\mathbb{G}$.*

Proof. We would like to prove that

$$\{[\mathbf{s}^\kappa]_2, \{[\langle \mathbf{s}^\kappa, \mathbf{v}_{L_\ell,i}\rangle]_2\}_{i\in\mathcal{HS}}\}_{\kappa\in[q_k]} \approx_c \{[\mathbf{s}^\kappa]_2, \{[v^\kappa_{L_\ell,i}]_2\}_{i\in\mathcal{HS}}\}_{\kappa\in[q_k]}$$

where q_k is the number of queries to $\mathsf{QKeyGen}$, $\mathbf{s}^\kappa \leftarrow \mathbb{Z}_p^k$, $\mathbf{v}_{L_\ell,i} \leftarrow \mathbb{Z}_p^k$ for $i \in \mathcal{HS}$ s.t. $\sum_{i\in\mathcal{HS}} \mathbf{v}_{L_\ell,i} + \sum_{i\in\mathcal{CS}} \sum_{j\in[n]\setminus\{i\}} (-1)^{j<i} \mathsf{PRF}^{\mathsf{K}_{i,j}}(L) = \mathbf{0}$, and $v^\kappa_{L_\ell,i} \leftarrow \mathbb{Z}_p$ for $i \in \mathcal{HS}$ and $\kappa \in [q_k]$ s.t. $\sum_{i\in\mathcal{HS}} v_{L_\ell,i} +$

$\sum_{i\in\mathcal{CS}}\langle \mathbf{s}^\kappa, \mathbf{v}_{L_\ell,i}\rangle = 0$. It is not hard to see that the following indistinguishability suffices to prove the above indistinguishability:

$$([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, \ldots, [\mathbf{A}\mathbf{m}_d]_2) \approx_c ([\mathbf{A}]_2, [\mathbf{r}_1]_2, \ldots, [\mathbf{r}_d]_2)$$

where $d > 1, n$ are any natural numbers, $\mathbf{c}$ is any vectors in $\mathbb{Z}_p^k$, $\mathbf{A} \leftarrow \mathbb{Z}_p^{n\times k}$, $\mathbf{m}_1, \ldots, \mathbf{m}_d \leftarrow \mathbb{Z}_p^k$ s.t. $\sum_{i\in[d]} \mathbf{m}_i = \mathbf{c}$, and $\mathbf{r}_1, \ldots, \mathbf{r}_d \leftarrow \mathbb{Z}_p^n$ s.t. $\sum_{i\in[d]} \mathbf{r}_i = \mathbf{A}\mathbf{c}$. It is easy to see that they are distributed the same if $n \le k$, so we consider the case $n > k$. The above relation can be rewritten as

$$\begin{aligned}&([\mathbf{A}]_2, [\mathbf{A}\mathbf{m}_1]_2, \ldots, [\mathbf{A}\mathbf{m}_{d-1}]_2, [\mathbf{A}\mathbf{c} - \sum_{i\in[d-1]} \mathbf{A}\mathbf{m}_i]_2)\\ &\approx_c ([\mathbf{A}]_2, [\mathbf{r}_1]_2, \ldots, [\mathbf{r}_{d-1}]_2, [\mathbf{A}\mathbf{c} - \sum_{i\in[d-1]} \mathbf{r}_i]_2)\end{aligned}$$

This is implied by the $d-1$-fold MDDH$_k$ assumption, which assert that

$$[(\mathbf{A}, \mathbf{A}\mathbf{m}_1, \ldots, \mathbf{A}\mathbf{m}_{d-1})]_2 \approx_c [(\mathbf{A}, \mathbf{r}_1, \ldots, \mathbf{r}_{d-1})]_2.$$

□

Lemma 5.4. *For all $\ell \in [q_L]$, we have $\mathsf{H}^\beta_{\ell,2} = \mathsf{H}^\beta_{\ell,3}$*

Proof. As we see above, Eq. (5.1) holds due to the query condition in Definition 5.1. Thus, $\{v_{L_\ell,i}\}_{i\in[\mathcal{HS}]}$ and $\{v_{L_\ell,i} + f_i(x^{1,\beta}_{i,L_\ell}) - f_i(x^{1,0}_{i,L_\ell})\}_{i\in[\mathcal{HS}]}$ are both randomly distributed in $\mathbb{Z}_p$ such that the summation of these is equal to $-\sum_{i\in\mathcal{CS}}\langle \mathbf{s}, \mathbf{v}_{L_\ell,i}\rangle$. □

Lemma 5.5. *For all $\ell \in [q_L]$, we have $\mathsf{H}^\beta_{\ell,3} \approx_c \mathsf{H}^\beta_\ell$ if aFE is partially function-hiding and the MDDH$_k$ holds in $\mathbb{G}$.*

Proof. This lemma can be proven in the same way as $\mathsf{H}^\beta_{\ell-1} \approx_c \mathsf{H}^\beta_{\ell,2}$. □

6 Dynamic Decentralized FE for Attribute Weighted Sums

In this section, we present a dynamic decentralized FE scheme for attribute weighted sums (DDFE for AWS). In Appendix B, we show how it can be captured in the context of dynamic MPFE.

6.1 Definition

Definition 6.1 (Dynamic Decentralized Functional Encryption). Let $\mathcal{ID}, \mathcal{K}, \mathcal{M}$ be an ID space, a key space, and a message space, respectively. $\mathcal{M}$ consists of a public part $\mathcal{M}_{\mathsf{pub}}$ and a private part $\mathcal{M}_{\mathsf{priv}}$. Let f be a function such that $f : \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{K})^i \times \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{M})^i \to \mathcal{Z}$. A DDFE scheme for f consists of five algorithms.

$\mathsf{Setup}(1^\lambda)$**:** It takes a security parameter 1^λ and outputs a public parameter PP. The other algorithms implicitly take PP.

$\mathsf{LSetup}(\mathsf{PP})$**:** It takes PP and outputs local public parameter PK_i and a master secret key MSK_i. The following three algorithms implicitly take PK_i.

$\mathsf{Enc}(\mathsf{MSK}_i, m)$**:** It takes MSK_i and $m \in \mathcal{M}$, and outputs a ciphertext CT_i.

$\mathsf{KeyGen}(\mathsf{MSK}_i, k)$**:** It takes MSK_i and $k \in \mathcal{K}$, and outputs a secret key SK_i.

$\mathsf{Dec}(\{\mathsf{SK}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{CT}_i\}_{i\in\mathcal{U}_M})$**:** It takes $\{\mathsf{SK}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}$ and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$ where $\mathcal{U}_K \subseteq \mathcal{ID}$ and $\mathcal{U}_M \subseteq \mathcal{ID}$ are any sets.

Correctness. A DDFE scheme for f is correct if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $\mathcal{U}_K \subseteq \mathcal{ID}$, $\mathcal{U}_M \subseteq \mathcal{ID}$, $\{i, k_i\}_{i\in\mathcal{U}_K} \in \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{K})^i$, $\{i, m_i\}_{i\in\mathcal{U}_M} \in \bigcup_{i\in\mathbb{N}}(\mathcal{ID}\times\mathcal{M})^i$, we have

$$\Pr\left[d = f(\{i,k_i\}_{i\in\mathcal{U}_K}, \{i,m_i\}_{i\in\mathcal{U}_M}) : \begin{array}{l} \mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{PK}_i, \mathsf{MSK}_i \leftarrow \mathsf{LSetup}(\mathsf{PP}) \\ \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}_i, m_i) \\ \mathsf{SK}_i \leftarrow \mathsf{KeyGen}(\mathsf{MSK}_i, k_i) \\ d = \mathsf{Dec}(\{\mathsf{SK}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}) \end{array}\right] = 1.$$

Note that we can consider the case where $\mathcal{U}_K$ and $\mathcal{U}_M$ are multisets as in the original definition in [CDSG+20]. However, we do not consider the case here since it induces ambiguity that can be also found in [CDSG+20][12]. We assume that $\mathbb{N}$ contains 0 here and $(\mathcal{ID}\times\mathcal{K})^0 = \{i, k_i\}_{i\in\emptyset} = \emptyset$. That is, $\mathcal{U}_K$ and $\mathcal{U}_M$ can be an empty set, which corresponds to the case where Dec does not take secret keys/ciphertexts as input.

Security. We define the security of DDFE as follows. A DDFE scheme is xx-yy-partially hiding (xx $\in$ {sel, adt}, yy $\in$ {sym, asym}) if for every stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\beta \leftarrow \mathcal{A}^{\mathsf{QHonestGen}(),\mathsf{QCor}(),\mathsf{QEnc}^\beta(),\mathsf{QKeyGen}()}(\mathsf{PP}) : \begin{array}{l} \beta \leftarrow \{0,1\} \\ \mathsf{PP} \leftarrow \mathsf{Setup}(1^\lambda) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Each oracle works as follows. For $i \in \mathcal{ID}$, $\mathsf{QHonestGen}(i)$ runs $(\mathsf{PK}_i, \mathsf{MSK}_i) \leftarrow \mathsf{LSetup}(\mathsf{PP})$ and returns PK_i. For i such that $\mathsf{QHonestGen}(i)$ was queried, the adversary can make the following queries: $\mathsf{QCor}(i)$ outputs MSK_i, $\mathsf{QEnc}^\beta(i, m^0, m^1)$ outputs $\mathsf{Enc}(\mathsf{MSK}_i, m^\beta)$, and $\mathsf{QKeyGen}(i, k)$ outputs $\mathsf{KeyGen}(\mathsf{MSK}_i, k)$. Note that m^β consists of the private elements m^β_{priv} and the public elements m_{pub}, respectively (we always require that $m^0_{\mathsf{pub}} = m^1_{\mathsf{pub}} = m_{\mathsf{pub}}$ as the public elements are not hidden in CT). Let $\mathcal{S}$ be the set of parties on which $\mathsf{QHonestGen}(i)$ is queried, $\mathcal{HS}$ be the set of parties on which the adversary has not queried QCor at the end of the game, and $\mathcal{CS} = \mathcal{S}\backslash\mathcal{CS}$. Then, the adversary's queries must satisfy the following conditions.

- There are no sequences $(\{i, k_i\}_{i\in\mathcal{U}_K}, \{i, m^0_i\}_{i\in\mathcal{U}_M}), \{i, m^1_i\}_{i\in\mathcal{U}_M})$ that satisfy all the conditions:
  - For all $i \in \mathcal{U}_K$, [$\mathsf{QKeyGen}(i, k_i)$ is queried] or [$i \in \mathcal{CS}$].
  - For all $i \in \mathcal{U}_M$, [$\mathsf{QEnc}^\beta(i, m^0_i, m^1_i)$ is queried] or [$m^0_i = m^1_i \in \mathcal{M}$ and $i \in \mathcal{CS}$].
  - $f(\{i, k_i\}_{i\in\mathcal{U}_K}, \{i, m^0_i\}_{i\in\mathcal{U}_M}) \neq f(\{i, k_i\}_{i\in\mathcal{U}_K}, \{i, m^1_i\}_{i\in\mathcal{U}_M})$.
- When xx = sel: the adversary first generates a set $\mathcal{S}$ of honest users in one shot. After that it makes the corruption, key generation, encryption queries in one shot to obtain $\{\mathsf{MSK}_i\}$, $\{\mathsf{KeyGen}(\mathsf{MSK}_i, k)\}$, $\{\mathsf{Enc}(\mathsf{EK}_i, m^\beta)\}$.
- When yy = sym: for $i \in \mathcal{CS}$, the queries $\mathsf{QEnc}^\beta(i, m^0, m^1)$ must satisfy $m^0 = m^1$[13].

We formally define DDFE for AWS as follows.

Definition 6.2 (DDFE for Attribute Weighted Sum). DDFE for AWS is a class of DDFE (Definition 6.1) where $\mathcal{ID} \subseteq \{0,1\}^*$, $\mathcal{K} = \bigcup_{S\subseteq\mathcal{ID}}(\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^S \times S$[14], $\mathcal{M} = \mathcal{X} \times 2^{\mathcal{ID}} \times \mathcal{L}$, where $\mathcal{X} = \bigcup_{i\in\mathbb{N}}(\mathbb{Z}_p^{n_0} \times \mathbb{Z}_p^{n_1})^i$, and supports the following functionality: Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The function f' is defined as follows: for $k_i = (\bar{f}_i, \mathcal{U}_{K,i}) \in \mathcal{K}$, where $\bar{f}_i = \{f_j\}_{j\in\mathcal{U}_{K,i}}$ and

[12] Concretely, when $\mathcal{U}_K$ is a multiset, and $i' \in \mathcal{U}_K$ has multiplicity 2, how to treat $k_{i'} \in \{k_i\}_{i\in\mathcal{U}_K}$ is unclear.

[13] The symmetric setting captures the case where MSK_i can be used to not only encrypt/key generation but also decryption/decoding of $\mathsf{CT}_i/\mathsf{SK}_i$.

[14] An element in $(\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^S \times S$ is of the form $(\{f_i\}_{i\in S}, S)$. We note that in more precise notation, $(\mathcal{F}^{\mathsf{ABP}}_{n_0,n_1})^S$ contains elements of the form $\{i, f_i\}_{i\in S}$, which itself carries information about S, but we explicitly add $\times S$, to keep the notation more intuitive.

$m_i = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j\in[N_i]}, \mathcal{U}_{M,i}, L_i) \in \mathcal{M}$ (here $\{\mathbf{z}_{i,j}\}_{j\in[N_i]}$ is the private part and $\{\mathbf{x}_{i,j}\}_{j\in[N_i]}, \mathcal{U}_{M,i}, L_i$ are the public parts of m_i),

$$f'(\{i,k_i\}_{i\in\mathcal{U}'_K}, \{i,m_i\}_{i\in\mathcal{U}'_M}) = \begin{cases} [\sum_{i\in\mathcal{U}'_K}\sum_{j\in[N_i]}\langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle]_T & \text{the condition below is satisfied} \\ \perp & \text{otherwise} \end{cases}$$

1. $\mathcal{U}'_K = \mathcal{U}'_M$ and $\forall\ i \in \mathcal{U}'_K$, $\mathcal{U}_{K,i} = \mathcal{U}_{M,i} = \mathcal{U}'_K$.
2. $\forall_{i,i'\in\mathcal{U}'_K}$, $\bar{f}_i = \bar{f}_{i'}$ and $L_i = L_{i'}$.

For a building block of DDFE for AWS, we use a class of DDFE called all-or-nothing encryption. Chotard *et al.* showed that sel-sym-IND-secure AoNE can be generically constructed from identity-based encryption [CDSG+20].

Definition 6.3 (All-or-nothing encryption (AoNE)). AoNE is a class of DDFE (Definition 6.1) where $\mathcal{ID} = \{0,1\}^*$, $\mathcal{M}_{\mathsf{priv}} = \{0,1\}^L$ for some $L \in \mathbb{N}$, $\mathcal{M}_{\mathsf{pub}} = 2^{\mathcal{ID}} \times \mathcal{L}$, $\mathcal{K} = \emptyset$, $\mathcal{Z} = \{0,1\}^*$. The function f is defined as, for $\mathcal{U}'_K \in 2^{\mathcal{ID}}$ and $\{m_i = (x_i, \mathcal{U}_{M,i}, L_{M,i})\}_{i\in\mathcal{U}'_M}$,

$$f(\{i\}_{i\in\mathcal{U}'_K}, \{i,m_i\}_{i\in\mathcal{U}'_M}) = \begin{cases} \{x_i\}_{i\in\mathcal{U}'_M} & \text{the condition below is satisfied} \\ \perp & \text{otherwise} \end{cases}$$

- $\forall i \in \mathcal{U}'_M, \mathcal{U}'_M = \mathcal{U}_{M,i}$.
- $\exists L_M \in \mathcal{L}, \forall i \in \mathcal{U}'_M, L_{M,i} = L_M$.

This means that KeyGen is unnecessary, and Dec works without taking secret keys as input in AoNE (recall that $\mathcal{U}'_K$ can be an empty set).

6.2 Construction

Let $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ be an FE scheme for AWSw/IP with the length of the random tape for aSetup being ℓ_a, $\mathsf{AoNE} = (\mathsf{anGSetup}, \mathsf{anLSetup}, \mathsf{anEnc}, \mathsf{anDec})$ be an all-or-nothing encryption scheme[15], $\mathsf{NIKE} = (\mathsf{nSetup}, \mathsf{nKeyGen}, \mathsf{nSharedKey})$ be a non-interactive key exchange scheme, $\{\mathsf{PRF}_1^{\mathsf{K}}\} : 2^{\mathcal{ID}} \times \mathcal{L} \to \mathbb{Z}_p^k$, $\{\mathsf{PRF}_2^{\mathsf{K}}\} : 2^{\mathcal{ID}} \to \{0,1\}^{\ell_s}$ be families of pseudorandom functions, with key space $\mathcal{K}_1, \mathcal{K}_2$, respectively and $\mathcal{ID}$ denotes an identity space and $H : \{0,1\}^* \to G_2^k$ is a hash function modeled as a random oracle. Our construction of DDFE for AWS is given in Figure 7

Correctness and Security. Firstly, we observe that if $\mathcal{U}_K = \mathcal{U}_M = \mathcal{U}$, $L_i = L_M$ for all $i \in \mathcal{U}$, where L_M is any label in $\mathcal{L}$[2] and $\{f_j\}_{j\in\mathcal{U}_{K,i}}$ is same in all the ciphertexts input to the decryption algorithm, then

- we have from the correctness of AoNE, $\widetilde{\mathsf{aSK}}_i = \mathsf{aSK}_i$ and $\widetilde{\mathsf{aCT}}_i = \mathsf{aCT}_i$.
- vector $\mathbf{s}$ computed by every user $i \in \mathcal{U}$ is same.

Then from the correctness of NIKE, $\mathsf{K}_{i,j,1} = \mathsf{K}_{j,i,1}$ and hence, $\sum_{i\in\mathcal{U}} \mathbf{v}_i = 0$. Hence, from the correctness of aFE,

$$\prod_{i\in\mathcal{U}} \mathsf{aDec}(\widetilde{\mathsf{aSK}}_i, \widetilde{\mathsf{aCT}}_i) = \prod_{i\in\mathcal{U}} \mathsf{aDec}(\mathsf{aSK}_i, \mathsf{aCT}_i) = \prod_{i\in\mathcal{U}} [\sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + \langle \mathbf{s}, \mathbf{v}_i\rangle]_T$$
$$= [\sum_{i\in\mathcal{U}}\sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}\rangle + \sum_{i\in\mathcal{U}} \langle \mathbf{s}, \mathbf{v}_i\rangle]_T = [\sum_{i\in\mathcal{U}}\sum_{j\in[N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i.j}\rangle]_T.$$

We argue security via the following theorem.

[15] We use AoNE to encrypt messages from two different spaces. So, either we can use two AoNE schemes with appropriate message spaces or can use padding to make the message spaces same. For simplicity, we present our construction with same AoNE scheme.

$\mathsf{GSetup}(1^\lambda)$**:** On input the security parameter 1^λ, the setup algorithm outputs PP as follows.

$$\mathsf{anPP} \leftarrow \mathsf{anGSetup}(1^\lambda),\ \mathsf{nPP} \leftarrow \mathsf{nSetup}(1^\lambda),\ \mathsf{PP} = (\mathsf{anPP}, \mathsf{nPP}).$$

$\mathsf{LSetup}(\mathsf{PP})$**:** On input PP, user $i \in \mathcal{ID}$ generates $(\mathsf{PK}_i, \mathsf{MSK}_i)$ via the setup algorithm as follows.

$$(\mathsf{nPK}_i, \mathsf{nSK}_i) \leftarrow \mathsf{nKeyGen}(\mathsf{nPP}),\ (\mathsf{anPK}_i, \mathsf{anMSK}_i) \leftarrow \mathsf{anLSetup}(\mathsf{anPP}),\ \mathsf{K}_{i,2} \leftarrow \mathcal{K}_2$$
$$\mathsf{PK}_i = (\mathsf{nPK}_i, \mathsf{anPK}_i),\ \mathsf{MSK}_i = (\mathsf{nSK}_i, \mathsf{anMSK}_i, \mathsf{K}_{i,2}).$$

$\mathsf{Enc}(\mathsf{MSK}_i, m)$**:** The encryption algorithm takes as input the public parameters PP, the master secret key MSK_i, and an input $m = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j \in N_i} \mathcal{U}_{M,i}, L_i)$ such that $i \in \mathcal{U}_{M,i}$ and outputs CT_i as follows.

$$\mathsf{rt}_i = \mathsf{PRF}_2^{\mathsf{K}_{i,2}}(\mathcal{U}_{M,i}),\ \mathsf{aMSK}_i = \mathsf{aSetup}(1^\lambda; \mathsf{rt}_i),\ \mathsf{K}_{i,j,1} \leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_i, \mathsf{nPK}_j)$$
$$\mathbf{v}_i = \sum_{\substack{j \in \mathcal{U}_{K,i} \\ i \neq j}} (-1)^{j<i} \mathsf{PRF}_1^{\mathsf{K}_{i,j,1}}(\mathcal{U}_{M,i}, L_i),\ \hat{x}_i = (\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{j \in N_i}, [\mathbf{v}_i, 0]_1),$$
$$\mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, \hat{x}_i) \tag{6.1}$$
$$\mathsf{anCT}_i \leftarrow \mathsf{anEnc}(\mathsf{anMSK}_i, (\mathsf{aCT}_i, \mathcal{U}_{M,i}, L_i)),\ \mathsf{CT}_i = (\mathsf{anCT}_i, \mathcal{U}_{M,i}, L_i). \tag{6.2}$$

$\mathsf{KeyGen}(\mathsf{MSK}_i, k)$**:** The key generation algorithm takes the master secret key MSK_i, and an input $k = (\{f_j\}_{j \in \mathcal{U}_{K,i}}, \mathcal{U}_{K,i})$ such that $i \in \mathcal{U}_{K,i}$ and outputs SK_i as follows.

$$\mathsf{rt}_i = \mathsf{PRF}_2^{\mathsf{K}_{i,2}}(\mathcal{U}_{K,i}),\ \mathsf{aMSK}_i = \mathsf{aSetup}(1^\lambda; \mathsf{rt}_i)$$
$$[\mathbf{s}]_2 = H(\{f_i\}_{i \in \mathcal{U}_{K,i}}, \mathcal{U}_{K,i}),\ \hat{f}_i = (f_i, [(\mathbf{s}, 0)]_2),\ \mathsf{aSK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, \hat{f}_i) \tag{6.3}$$
$$\mathsf{anCT}_i \leftarrow \mathsf{anEnc}(\mathsf{anMSK}_i, (\mathsf{aSK}_i, \mathcal{U}_{K,i}, \{f_j\}_{j \in \mathcal{U}_{K,i}})),\ \mathsf{SK}_i = (\mathsf{anCT}_i, \mathcal{U}_{K,i}, \{f_j\}_{j \in \mathcal{U}_{K,i}}). \tag{6.4}$$

$\mathsf{Dec}(\{\mathsf{SK}_i\}_{i \in \mathcal{U}_K}, \{\mathsf{CT}_i\}_{i \in \mathcal{U}_M})$**:** The decryption algorithm takes as input the public parameters PP, secret keys $\{\mathsf{SK}_i\}_{i \in \mathcal{U}_K}$, ciphertexts $\{\mathsf{CT}_i\}_{i \in \mathcal{U}_M}$ such that $\mathcal{U} = \mathcal{U}_K = \mathcal{U}_M$ and outputs d as follows. Parse $\mathsf{SK}_i = (\mathsf{anCT}_i, \mathcal{U}_{K,i}, \{f_j\}_{j \in \mathcal{U}_{K,i}})$ and $\mathsf{CT}_i = (\mathsf{anCT}'_i, \mathcal{U}_{M,i}, L_i)$. Compute

$$\{\widetilde{\mathsf{aSK}_i}\}_{i \in \mathcal{U}} = \mathsf{anDec}(\{\mathsf{anCT}_i\}_{i \in \mathcal{U}}),\ \{\widetilde{\mathsf{aCT}_i}\}_{i \in \mathcal{U}} = \mathsf{anDec}(\{\mathsf{anCT}'_i\}_{i \in \mathcal{U}}),$$
$$[d]_T = \prod_{i \in \mathcal{U}} \mathsf{aDec}(\widetilde{\mathsf{aSK}_i}, \widetilde{\mathsf{aCT}_i}).$$

Fig 7. Dynamic Decentralized FE for AWS

Theorem 6.1. *If* $\{\mathsf{PRF}_1^{\mathsf{K}}\}, \{\mathsf{PRF}_2^{\mathsf{K}}\}$ *are families of pseudorandom functions,* NIKE *is IND-secure,* AoNE *is sel-sym-IND-secure, the MDDH$_k$ assumption holds in* $\mathbb{G}$*, and* aFE *is function-hiding, then our AWS-DDFE scheme is sel-sym-partially-hiding in the random oracle model as per Definition 6.1.*

Proof. Let $\mathcal{S}$ be the set of parties generated by $\mathsf{QHonestGen}$ queries. Let $\mathcal{HS} \subseteq \mathcal{S}$ be the set of uncorrupted parties and $\mathcal{CS} = \mathcal{S} \backslash \mathcal{HS}$. We prove the theorem via a series of hybrids, which are defined as follows.

H_s^β**:** This is the original game. In particular, in response to $\mathsf{QEnc}^\beta(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ and $\mathsf{QKeyGen}(i, \{f_j\}_{j \in \mathcal{U}_K}, \mathcal{U}_K)$, where $x_i^b = \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}^b\}_{j \in [N_i]}$ for $b \in \{0,1\}$, the challenger sets

$$\hat{x}_i = (x_i^\beta, [\mathbf{v}_i, 0]_1),\quad \hat{f}_i = (f_i, [\mathbf{s}, 0]_2),$$

(in eqs. (6.1) and (6.3), respectively). Vectors $\mathbf{v}_i$ and $\mathbf{s}$ are computed as desribed in the construction.

H_1^β**:** In this hybrid, the challenger samples rt_i randomly instead of computing using PRF_2. Indistinguishability between H_s^β and H_1^β follows from the security of PRF_2.

H_2^β**:** We say an encryption query on $(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ is incomplete, if there exists $i' \in \mathcal{U}_M$ such that $i' \in \mathcal{HS}$ and no encryption query of the form $(i', \star, \star, \mathcal{U}_M, L_M)$ is made. In this hybrid, in response to all the incomplete encryption queries, anCT_i is computed as $\mathsf{anEnc}.(0, \mathcal{U}_M, L_M)$ (eq. (6.2)). The indistinguishability between H_1^β and H_2^β follows from the security of AoNE.

H_3^β**:** We say a key query on $(i, \bar{f} = \{f_j\}_{j \in \mathcal{U}_K}, \mathcal{U}_K)$ is incomplete if there exist $i' \in \mathcal{U}_K$ such that $i' \in \mathcal{HS}$ and there is no key query of the form $(i', \bar{f}, \mathcal{U}_K)$. In this hybrid, for all the incomplete key queries anCT_i encrypts 0 (eq. (6.4)). The indistinguishability between H_2^β and H_3^β follows from the security of AoNE.

H_f^β**:** In this hybrid, for all the complete encryption queries of the form $(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ with $i \in \mathcal{HS}$, the challenger sets $\hat{x}_i = (x_i^0, [\mathbf{v}_i, 0]_1)$. We note that the adversary has zero advantage in this hybrid because its view is independent of β (recall that for $i \in \mathcal{CS}$, $x_i^0 = x_i^1$.). We show that H_f^β is indistinguishable from H_3^β in Lemma 6.1

Lemma 6.1. *If* $\{\mathsf{PRF}_1^\mathsf{K}\}$ *is a family of pseudorandom functions,* NIKE *is* IND*-secure, the MDDH*$_k$ *assumption holds in* $\mathbb{G}$*, and* aFE *is partially function-hiding, then* $\mathsf{H}_3^\beta \approx \mathsf{H}_f^\beta$ *in the random oracle model.*

Proof. To prove the lemma, we consider the following sub hybrids between H_3^β and H_f^β. Let q_u be the total number of ID sets with complete encryption queries. Let $\{\mathcal{U}_1, \ldots, \mathcal{U}_{q_u}\}$ be some fixed ordering on the ID sets from complete encryption queries and let q'_u be the upper bound on q_u. Then define sub hybrid $\widehat{\mathsf{H}}_j^\beta$ as follows

$\widehat{\mathsf{H}}_j^\beta$**:** (for $j \in \{0\} \cup [q'_u]$). This hybrid is same as H_3^β except that for every complete encryption query of the form $(i, x_i^0, x_i^1, \mathcal{U}_M, L_M)$ such that $i \in \mathcal{HS}$, the challenger sets

$$\hat{x}_i = \begin{cases} (x_i^0, [\mathbf{v}_i, 0]_1) & \text{if } \mathcal{U}_M \in \{\mathcal{U}_1, \ldots, \mathcal{U}_j\} \\ (x_i^\beta, [\mathbf{v}_i, 0]_1) & \text{if } \mathcal{U}_M \in \{\mathcal{U}_{j+1}, \ldots, \mathcal{U}_{q_u}\}, \end{cases}$$

where $\mathcal{U}_j = \{\perp\}$ for $j > q_u$. We observe that $\widehat{\mathsf{H}}_0^\beta = \mathsf{H}_3^\beta$ and $\widehat{\mathsf{H}}_{q'_u}^\beta = \mathsf{H}_f^\beta$. So, now we need to show that for all $j \in [q'_u]$, $\widehat{\mathsf{H}}_{j-1}^\beta \approx \widehat{\mathsf{H}}_j^\beta$.

To show this, we let $\{L_{\mathcal{U}_j}^1, \ldots, L_{\mathcal{U}_j}^v\}$ be the set of labels used in complete encryption queries of the form $(\star, \star, \star, \mathcal{U}_j, \star)$. Let $v \leq q_L$. Then define the following sub hybrids:

$\widehat{\mathsf{H}}_{j-1,0}^\beta$**:** Same as $\widehat{\mathsf{H}}_{j-1}^\beta$.

$\widehat{\mathsf{H}}_{j-1,\kappa}^\beta$**:** (for $\kappa \in [q_L]$). Same as $\widehat{\mathsf{H}}_{j-1}^\beta$ except that for every complete encryption query of the form $(i, x_i^0, x_i^1, \mathsf{U}_j, L)$, for $i \in \mathcal{HS}$,

$$\hat{x}_i = \begin{cases} (x_i^0, [\mathbf{v}_i, 0]_1) & \text{if } L \in \{L_{\mathcal{U}_j}^1, \ldots, L_{\mathcal{U}_j}^\kappa\} \\ (x_i^\beta, [\mathbf{v}_i, 0]_1) & \text{if } L \in \{L_{\mathcal{U}_j}^{\kappa+1}, \ldots, L_{\mathcal{U}_j}^v\} \end{cases}$$

We observe that $\widehat{\mathsf{H}}_{j-1,q_L}^\beta = \widehat{\mathsf{H}}_j^\beta$. So now, we need to show that $\widehat{\mathsf{H}}_{j-1,\kappa-1}^\beta \approx \widehat{\mathsf{H}}_{j-1,\kappa}^\beta$, for all $\kappa \in [q_L]$. For this, we further define following sub hybrids between $\widehat{\mathsf{H}}_{j-1,\kappa-1}^\beta$ and $\widehat{\mathsf{H}}_{j-1,\kappa}^\beta$:

Let $\mathcal{U}_j^{\mathcal{HS}} = \mathcal{U}_j \cap \mathcal{HS} = \{u_1, \ldots, u_w\}$ and w' be an upper bound on w. Define

$\bar{\mathsf{H}}_\eta^\beta$**:** (for $\eta \in [w']$). Same as $\widehat{\mathsf{H}}_{j-1,\kappa-1}^\beta$, except that for each complete encryption query $\mathsf{QEnc}^\beta(u_i, x_{u_i}^0, x_{u_i}^1, \mathcal{U}_j, L_{\mathcal{U}_j}^\kappa)$ and complete key query $\mathsf{QKeyGen}(u_i, \{f_j\}_{j \in \mathcal{U}_j}, \mathcal{U}_j)$, $\hat{x}_{u_i}$ and $\hat{f}_{u_i}$, respectively, are set as follows:

$$\hat{x}_{u_i} = \begin{cases} (x_{u_i}^0, [\mathbf{v}_{u_i}, 0]_1) & \text{if } i \leq \eta \\ (x_{u_i}^\beta, [\mathbf{v}_{u_i}, 0]_1) & \text{if } \eta < i < w \\ (x_{u_i}^\beta, [\mathbf{v}_{u_i}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, 0]_2) & \text{if } i < w \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}]_2) & \text{if } i = w \end{cases}$$

Here, $\Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}} = f_{u_l}(x^{1,\beta}_{u_l}) - f_{u_l}(x^{1,0}_{u_l})$, where 1 in the superscript indicates the first QEnc^{β} query of the form $(u_l, \star, \star, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$. We have, from the admissibility conditions,

- Let $q_{c,u_l,\mathcal{U}_j,L^{\kappa}_{\mathcal{U}_j}}$ be the number of encryption queries of the form $(u_l, \star, \star, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$, then $f_{u_l}(x^{\tau,\beta}_{u_l}) - f_{u_l}(x^{\tau,0}_{u_l}) = \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}$ for all $\tau \in [q_{c,u_l,\mathcal{U}_j,L^{\kappa}_{\mathcal{U}_j}}]$, where τ denotes the sequence number of the query of this form (see proof of Lemma 5.2).
- $\sum_{u_l\in(\mathcal{HS}\cap\mathcal{U}_j)} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}} = 0$.

Now we argue the indistinguishability of the sub hybrids. Firstly, we observe the following:

1. $\widehat{\mathsf{H}}^{\beta}_{j-1,\kappa-1} \approx \bar{\mathsf{H}}^{\beta}_{0}$: The only difference between the two hybrids is that for encryption query of the form $\mathsf{QEnc}^{\beta}(u_w, x^0_{u_w}, x^1_{u_w}, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$, $\hat{x}_{u_w} = (x^{\beta}_{u_w}, [\mathbf{v}_{u_w}, 0]_1)$ in the former and $\hat{x}_{u_w} = (x^{\beta}_{u_w}, [\mathbf{v}_{u_w}, 1]_1)$ in the latter hybrid. Note that $\hat{f}_{u_w}$ for any key queries of the form $(u_w, \{f_j\}_{j\in[\mathcal{U}_j]}, \mathcal{U}_j)$ is of the form $(f_{u_w}, [\mathbf{s}, 0]_2)$ (notice the last bit being 0) in both the hybrids. Hence, the two hybrids are indistinguishable due to partially function-hiding security of aFE.
2. Similarly, $\bar{\mathsf{H}}^{\beta}_{w'} \approx \widehat{\mathsf{H}}^{\beta}_{j-1,\kappa}$ from aFE security.

So, all that is left is to show that $\bar{\mathsf{H}}^{\beta}_{\eta-1} \approx \bar{\mathsf{H}}^{\beta}_{\eta}$. For this, we first note that the two hybrids differ only in the values of $\hat{x}_{u_\eta}$ and $\hat{f}_{u_w}$ as follows:

In $\bar{\mathsf{H}}^{\beta}_{\eta-1}$:

$$\hat{x}_{u_i} = \begin{cases} (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 0]_1) & \text{if } i = \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 1]_1) & \text{if } i = w \end{cases}, \quad \hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, 0]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta-1]} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}]_2) & \text{if } i = w \end{cases}$$

In $\bar{\mathsf{H}}^{\beta}_{\eta}$:

$$\hat{x}_{u_i} = \begin{cases} (x^{0}_{u_i}, [\mathbf{v}_{u_i}, 0]_1) & \text{if } i = \eta \\ (x^{\beta}_{u_i}, [\mathbf{v}_{u_i}, 1]_1) & \text{if } i = w \end{cases}, \quad \hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, 0]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^{\beta}_{u_l, L^{\kappa}_{\mathcal{U}_j}}]_2) & \text{if } i = w \end{cases}$$

To show indistinguishability, we consider sub hybrids with the following sequence of changes in $\hat{x}_{u_i}$ and $\hat{f}_{u_i}$ for $u_i \in \mathcal{U}^{\mathcal{HS}}_j$.

$\bar{\mathsf{H}}^{\beta}_{\eta-1,1}$**:** For every complete QEnc^{β} query, sample $K_{u_\eta,u_w,1} (= K_{u_w,u_\eta,1})$ randomly instead of computing from $\mathsf{nSharedKey}$[16]. Indistinguishability from $\bar{\mathsf{H}}^{\beta}_{\eta-1}$ follows from the security of NIKE.

$\bar{\mathsf{H}}^{\beta}_{\eta-1,2}$**:** For any complete encryption query of the form $(u_i, \star, \star, \mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})$, the computation of $\mathbf{v}_{u_\eta}$ and $\mathbf{v}_{u_w}$ use random value in place of $\mathsf{PRF}_1^{K_{u_\eta,u_w,1}(\mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})}$. Thus, vectors $\mathbf{v}_{u_\eta}$ and $\mathbf{v}_{u_w}$ changes from

$$\mathbf{v}_{u_\eta} = \sum_{i\in\mathcal{U}_j, i\neq u_\eta} (-1)^{i<u_\eta} \mathsf{PRF}^{K_{u_\eta,i,1}(\mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})}, \text{ to}$$

$$\mathbf{v}_{u_\eta} = \sum_{i\in\mathcal{U}_j, i\notin\{u_\eta, u_w\}} (-1)^{i<u_\eta} \mathsf{PRF}^{K_{u_\eta,i,1}(\mathcal{U}_j, L^{\kappa}_{\mathcal{U}_j})} \underline{+\mathbf{t}_{u_\eta,u_w}}$$

$$\mathbf{v}_{u_w} = \sum_{i\in\mathcal{U}_j, i\neq u_w} (-1)^{i<w} \mathsf{PRF}^{K_{u_w,i,1}(\mathcal{U}_\ell, L^{\kappa}_{\mathcal{U}_j})}, \text{ to}$$

$$\mathbf{v}_{u_w} = \sum_{i\in\mathcal{U}_j, i\notin\{u_\eta, u_w\}} (-1)^{i<u_w} \mathsf{PRF}^{K_{u_w,i,1}(\mathcal{U}_\ell, L^{\kappa}_{\mathcal{U}_j})} \underline{-\mathbf{t}_{u_\eta,u_w}}$$

[16] this change will happen for all the ID sets, since $K_{u_\eta,u_w,1}$ does not depend on the ID set or the label.

where $\mathbf{t}_{u_\eta,u_w}$ is chosen randomly. Indistinguishability from the previous sub hybrid follows from the security of PRF_1.

$\bar{\mathsf{H}}^\beta_{\eta-1,3}$**:** Change $\hat{x}_{u_\eta}$, $\hat{x}_{u_w}$, $\hat{f}_{u_\eta}$ and $\hat{f}_{u_w}$ as:

$$\hat{x}_{u_i} = \begin{cases} (x^\beta_{u_i}, [\mathbf{v}_{u_i} \underline{+\mathbf{t}_{u_\eta,u_w}}, 1]_1) & \text{if } i = \eta \\ (x^\beta_{u_i}, [\mathbf{v}_{u_i} \underline{-\mathbf{t}_{u_\eta,u_w}}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, \underline{-\langle \mathbf{s}, \mathbf{t}_{u_\eta,u_w} \rangle}]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \underline{\sum_{l\in[\eta-1]} \Delta^\beta_{u_l,L^\kappa_{\mathcal{U}_j}} + \langle \mathbf{s}, \mathbf{t}_{u_\eta,u_w} \rangle}]_2) & \text{if } i = w \end{cases}$$

The indistinguishability follows from the partially function-hiding property of aFE.

$\bar{\mathsf{H}}^\beta_{\eta-1,4}$**:** Replace $\langle \mathbf{s}, \mathbf{t}_{u_\eta,u_w} \rangle$ with random t_{u_η,u_w}.

$$\hat{\mathbf{x}}_{u_i} = \begin{cases} (x^\beta_{u_i}, [\mathbf{v}_{u_i} + \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = \eta \\ (x^\beta_{u_i}, [\mathbf{v}_{u_i} - \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, \underline{-t_{u_\eta,u_w}}]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta-1]} \Delta^\beta_{u_l,L^\kappa_{\mathcal{U}_j}} + \underline{t_{u_\eta,u_w}}]_2) & \text{if } i = w \end{cases}$$

Indistinguishability between $\bar{\mathsf{H}}^\beta_{\eta-1,3}$ and $\bar{\mathsf{H}}^\beta_{\eta-1,4}$ follows from the MDDH_k assumption. In more detail, let $\bar{f}^1, \ldots, \bar{f}^{q_k}$ be the functions for which the adversary issues complete key queries of the form $(\star, \star, \mathcal{U}_j)$ queries and let $\mathbf{s}^1, \ldots, \mathbf{s}^{q_k}$ be the corresponding $\mathbf{s}$ vectors (recall that these are computed from the hash function modeled as a random oracle). Then, to argue indistinguishability between the two hybrids, we need to show

$$\{\mathbf{s}^\tau, \langle \mathbf{s}^\tau, \mathbf{t}_{u_\eta,u_w} \rangle\}_{\tau\in[q_k]} \approx \{t^\tau_{u_\eta,u_w}\}_{\tau\in[q_k]},$$

which follows directly from the MDDH_k assumption.

$\bar{\mathsf{H}}^\beta_{\eta-1,5}$**:** Implicitly set $t_{u_\eta,u_w} = t'_{u_\eta,u_w} + \Delta_{u_\eta,L^\kappa_{\mathcal{U}_j}}$. That is,

$$\hat{\mathbf{x}}_{u_i} = \begin{cases} (x^\beta_{u_i}, [\mathbf{v}_{u_i} + \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = \eta \\ (x^\beta_{u_i}, [\mathbf{v}_{u_i} - \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, \underline{-t'_{u_\eta,u_w} - \Delta_{u_\eta,L^\kappa_{\mathcal{U}_j}}}]_2) & \text{if } i = \eta \\ (f_{u_w}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^\beta_{u_l,L^\kappa_{\mathcal{U}_j}} + \underline{t'_{u_\eta,u_w}}]_2) & \text{if } i = w \end{cases}$$

$\bar{\mathsf{H}}^\beta_{\eta-1,6}$**:** Change $\hat{x}_{u_\eta}$ and $\hat{f}_\eta$ as

$$\hat{\mathbf{x}}_{u_i} = \begin{cases} (x^0_{u_i}, [\mathbf{v}_{u_i} + \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = \eta \\ (x^\beta_{u_i}, [\mathbf{v}_{u_i} - \mathbf{t}_{u_\eta,u_w}, 1]_1) & \text{if } i = w \end{cases}$$

$$\hat{f}_{u_i} = \begin{cases} (f_{u_i}, [\mathbf{s}, \underline{-t'_{u_\eta,u_w}}]_2) & \text{if } i = \eta \\ (f_{u_i}, [\mathbf{s}, \sum_{l\in[\eta]} \Delta^\beta_{u_l,L^\kappa_{\mathcal{U}_j}} + t'_{u_\eta,u_w}]_2) & \text{if } i = w \end{cases}$$

Indistinguishability follows from the partially function hiding property of aFE.

Now, undo the changes in previous steps to get $\bar{\mathsf{H}}^\beta_\eta$.

References

ABDP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.

ABG19. Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Heidelberg, December 2019.

ABV+12. Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *LNCS*, pages 280–297. Springer, Heidelberg, May 2012.

ACF+18. Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Heidelberg, August 2018.

ACF+20. Shweta Agrawal, Michael Clear, Ophir Frieder, Sanjam Garg, Adam O'Neill, and Justin Thaler. Ad hoc multi-input functional encryption. In Thomas Vidick, editor, *ITCS 2020*, volume 151, pages 40:1–40:41. LIPIcs, January 2020.

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 467–497. Springer, Heidelberg, December 2020.

AFV11. Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 21–40. Springer, Heidelberg, December 2011.

AGRW17. Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 601–626. Springer, Heidelberg, April / May 2017.

AGT21. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 224–255. Springer, Heidelberg, November 2021.

AGT22. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *TCC*, 2022.

AGW20. Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 685–716. Springer, Heidelberg, August 2020.

AYY22. Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In *CRYPTO*, 2022.

BGG+14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Heidelberg, May 2014.

BSW07. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society Press, May 2007.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

CDG+18a. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Heidelberg, December 2018.

CDG+18b. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. https://eprint.iacr.org/2018/1021.

CDSG+20. Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 747–775. Springer, Heidelberg, August 2020.

Cha07. Melissa Chase. Multi-authority attribute based encryption. In Salil P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 515–534. Springer, Heidelberg, February 2007.

CSW21. Michele Ciampi, Luisa Siniscalchi, and Hendrik Waldner. Multi-client functional encryption for separable functions. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 724–753. Springer, Heidelberg, May 2021.

DOT18. Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part II*, volume 10770 of *LNCS*, pages 245–277. Springer, Heidelberg, March 2018.

DP21. Pratish Datta and Tapas Pal. (Compact) adaptively secure FE for attribute-weighted sums from k-lin. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 434–467. Springer, Heidelberg, December 2021.

EHK+17. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *Journal of Cryptology*, 30(1):242–288, January 2017.

FFMV23. Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multi-key and multi-input predicate encryption from learning with errors. In *Eurocrypt*, 2023.

GGG+14a. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.

GGG+14b. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In *Eurocrypt*, 2014.

GPSW06. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 89–98. ACM Press, October / November 2006. Available as Cryptology ePrint Archive Report 2006/309.

GVW12. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 162–179. Springer, Heidelberg, August 2012.

GVW13. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 545–554. ACM Press, June 2013.

GVW15. Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 503–523. Springer, Heidelberg, August 2015.

IW14. Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.

KDK11. Klaus Kursawe, George Danezis, and Markulf Kohlweiss. Privacy-friendly aggregation for the smart-grid. In Simone Fischer-Hübner and Nicholas Hopper, editors, *PETS 2011*, volume 6794 of *LNCS*, pages 175–191. Springer, Heidelberg, July 2011.

KSW08. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008.

LL20a. Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 247–277. Springer, Heidelberg, May 2020.

LL20b. Huijia Lin and Ji Luo. Succinct and adaptively secure ABE for ABP from k-lin. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 437–466. Springer, Heidelberg, December 2020.

NPP22. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access control. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022*, volume 13791, pages 95–125. Springer, 2022.

SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer, Heidelberg, May 2005.

Tom19. Junichi Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 459–488. Springer, Heidelberg, December 2019.

Wat12. Brent Waters. Functional encryption for regular languages. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 218–235. Springer, Heidelberg, August 2012.

Wee17a. Hoeteck Wee. Attribute-hiding predicate encryption in bilinear groups, revisited. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.

Wee17b. Hoeteck Wee. Attribute-Hiding Predicate Encryption in Bilinear Groups, Revisited. In *TCC*, 2017.

Work	Parties	EK Cor.	Label	(Pub, Pri) CT	Key	Functionality	Function classes of f, g, h
AB-FE [ACGU20]	1	✓	N/A	$(x, \mathbf{z})$	$(f, \mathbf{c})$	$f(x) \cdot \langle \mathbf{z}, \mathbf{c} \rangle$	$f \in$ MSPs
FE for AWS [AGW20]	1	✓	N/A	$(\{\mathbf{x}_j\}_j, \{\mathbf{z}_j\}_j)$	f	$\sum_{j \in [N]} f(\mathbf{x}_j)^\top \mathbf{z}_j$	$f \in$ ABPs
MIFE [AGT22]	n	✓	×	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \otimes \mathbf{z} \rangle$	N/A
MIFE [AGT22]	n	×	✓	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \otimes \mathbf{z} \rangle$	N/A
AB-MIFE [ACGU20]	n	✓	×	$(\bot, \mathbf{z}_i)$	$\{y_i, \mathbf{c}_i\}_{i \in S}$	$f(\{y_i\}_{i \in S}) \cdot \sum_{i \in S} \langle \mathbf{z}_i, \mathbf{c}_i \rangle$	$f(\{y_i\}) = \bigwedge_{i \in S} g_i(y_i)$ fixed $g_i \in$ MSPs
AB-MIFE, §4,4.2	n	✓	×	$((\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_j), \{\mathbf{z}_{i,j}\}_j)$	$\{g_i, h_i\}_{i \in [n]}$	$f(\mathbf{y}) \cdot \sum_{i \in [n]} \sum_{j \in [N_i]} h_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$	$f(\mathbf{y}) = \bigwedge_i (g_i(\mathbf{y}_i) = 0)$ $g_i, h_i \in$ ABPs
MCFE [CDG+18b, ABG19]	n	✓	✓	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \rangle$	N/A
AB-MCFE [NPP22]	n	✓	OT	$(x_i, \mathbf{z}_i)$	$\{g_i, \mathbf{c}_i\}$	$f(\{x_i\}) \cdot \langle \mathbf{c},\ \mathbf{z} \rangle$	$f(\{x_i\}) = \bigwedge_i g_i(x_i)$ $g_i \in$ LSS
MCFE, §5	n	✓	✓	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i \in [n]}$	$\sum_{i \in [n]} \sum_{j \in [N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$	$f_i \in$ ABPs
DDFE, [CDSG+20, AGT21]	Unbdd n	✓	✓	$(\bot, \mathbf{z}_i)$	$\mathbf{c}$	$\langle \mathbf{c},\ \mathbf{z} \rangle$	N/A
DDFE, §6	Unbdd n	✓	✓	$(\{\mathbf{x}_{i,j}\}_j, \{\mathbf{z}_{i,j}\}_j)$	$\{f_i\}_{i \in S}$	$\sum_{i \in S} \sum_{j \in [N_i]} f_i(\mathbf{x}_{i,j})^\top \mathbf{z}_{i,j}$	$f_i \in$ ABPs

Table 3. Prior state of the art and our results. We do not consider function hiding here. Above, we denote $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$, $\mathbf{z} = (\mathbf{z}_1, \ldots, \mathbf{z}_n)$ or $\mathbf{z} = (\mathbf{z}_i)_{i \in S}$. $S \subseteq [n]$ is some subset of authorized users for a given key. EK Cor. refers to whether an adversary is allowed to obtain encryption keys in the security game. Label refers to the capability of labeling functionality that restricts decryption such that it is allowed only when all labels are equal. OT in label means that each label can be used only one time per input. MSPs/ABPs/LSS stand for monotone span programs/arithmetic branching programs/linear secret sharing. MCFE in a stronger (resp. weaker) notion corresponds to MIFE that satisfies EK Cor. and Label (resp. One-time label).

A Detailed Comparison with Prior Work

We summarize prior works in Table 3.

B Multi-Party Functional Encryption

In this paper, we use many classes of functional encryption (FE) such as attribute-based encryption, secret-key functional encryption, multi-input encryption, etc. To capture various notions of FE, Agrawal, Goyal, and Tomida proposed a notion called multi-party functional encryption (MPFE) [AGT21]. The following definition is verbatim from [AGT21].

Definition B.1 (Multi-Party Functional Encryption). Let n_x be the number of ciphertext inputs and n_y be the number of key inputs. Let $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ be the space of ciphertext inputs and $\mathcal{Y} = \mathcal{Y}_{\mathsf{pub}} \times \mathcal{Y}_{\mathsf{priv}}$ be the space of key inputs. We define two aggregation functions as $\mathsf{Agg}_x : \mathcal{X}^{n_x} \to \mathcal{X}^*$, and $\mathsf{Agg}_y : \mathcal{Y}^{n_y} \to \mathcal{Y}^*$.

An MPFE scheme is defined as a tuple of 4 algorithms/protocols $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. To suitably capture existing primitives, we define our Setup algorithm/protocol to run in three modes, described next.

Setup modes. The Setup algorithm/protocol can be run in different modes: central, local, or interactive. For $\mathsf{mode} \in \{\mathsf{central}, \mathsf{local}, \mathsf{interactive}\}$, consider the following.

central: Here the Setup algorithm is run by one trusted third party which outputs the master secret keys and encryption keys for all users in the system.

local: Here it is run independently by different parties without any interaction, and each party outputs its own encryption key and/or master secret key.

interactive: Here it is an interactive protocol run by a set of users, at the end of which, each user has its encryption key and/or master secret key. We note that these keys may be correlated across multiple users.

A multi-party functional encryption (MPFE) consists of the following:

$\mathsf{Setup}\left(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y\right)$**:** This algorithm/protocol can be executed in any one of the three modes described above.Given input the security parameter, number of ciphertext inputs n_x, number of key inputs n_y and two aggregation functions Agg_x, Agg_y as defined above, this algorithm outputs a set of encryption keys $\{\mathsf{EK}_i\}_{i\le n_x}$, master secret keys $\{\mathsf{MSK}_i\}_{i\le n_y}$ and public key PK.

$\mathsf{Enc}\left(\mathsf{PK}, \mathsf{EK}, i, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})\right)$**:** Given input the public key PK, an encryption key EK, user index $i \in [n_x]$, an input $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})$, this algorithm outputs a ciphertext CT_x.

$\mathsf{KeyGen}\left(\mathsf{PK}, \mathsf{MSK}, j, y = (y_{\mathsf{pub}}, y_{\mathsf{priv}})\right)$**:** Given input the public key PK, a master secret key MSK, user index $j \in [n_y]$ and a function input $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}})$, this algorithm outputs a secret key SK_y.

$\mathsf{Dec}\left(\mathsf{PK}, \{\mathsf{SK}_j\}_{j\le n_y}, \{\mathsf{CT}_i\}_{i\le n_x}\right)$**:** Given input the public key PK, a set of secret keys $\{\mathsf{SK}_j\}_{j\le n_y}$ and a set of ciphertexts $\{\mathsf{CT}_i\}_{i\le n_x}$, this algorithm outputs a value z or $\perp$.

We remark that in the *local* setup mode, it will be helpful to separate the setup algorithm into a global setup, denoted by Gsetup along with a local setup, denoted by Lsetup, where the former is used only to generate common parameters of the system, such as group descriptions and such.

Correctness. We say that an MPFE scheme is *correct* if, $\forall (n_x, n_y) \in \mathbb{N}^2$, ciphertext inputs $x_i \in \mathcal{X}$ for $i \in [n_x]$, key inputs $y_j \in \mathcal{Y}$ for $j \in [n_y]$, message and function aggregation circuits Agg_x and Agg_y, it holds that:

$$\Pr\left[\begin{array}{l} \left(\mathsf{PK}, \{\mathsf{EK}_i\}, \{\mathsf{MSK}_j\}\right) \leftarrow \mathsf{Setup}(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y) \\ \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{PK}, \mathsf{EK}_i, i, x_i)\ \forall i \in [n_x] \\ z = z' : \mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}_j, j, y_j)\ \forall j \in [n_y] \\ z \leftarrow \mathsf{Dec}\left(\mathsf{PK}, \{\mathsf{SK}_j\}_{j\le n_y}, \{\mathsf{CT}_i\}_{i\le n_x}\right) \\ z' = \mathcal{U}\left(\mathsf{Agg}_x(\{x_i\}), \mathsf{Agg}_y(\{y_j\})\right) \end{array}\right] = 1.$$

Recall that $\mathcal{U}$ is the universal circuit with appropriate input and output size.

Indistinguishability based security. Next, we define security of MPFE. The security definition is modelled in a similar fashion to MIFE security [GGG+14a, §2.2] while taking into account corruption queries.

For any choice of parameters λ, n_x, n_y, aggregation functions $\mathsf{Agg}_x, \mathsf{Agg}_y$, and master keys $\mathsf{K} = (\mathsf{PK}, \{\mathsf{EK}_i\}_{i\in[n_x]}, \{\mathsf{MSK}_j\}_{j\in[n_y]}) \leftarrow \mathsf{Setup}(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y)$, we define the following list of oracles:

$\mathsf{QCor}^{\mathsf{K}}(\cdot)$, upon a call to this oracle for any $i \in [n_x]$ or $j \in [n_y]$, the adversary gets the corresponding encryption key EK_i or master secret key MSK_j. In the case of a local setup, the adversary could instead also supply the oracle with adversarially generated keys for the corresponding user; whereas in case of an interactive setup, the adversary could simulate the behavior of the queried user index in the setup protocol. (Let $\mathcal{S}_x \subseteq [n_x]$ and $\mathcal{S}_y \subseteq [n_y]$ denote the set of user indices for which the corresponding encryption and master keys have been corrupted.)

$\mathsf{QEnc}^{\mathsf{K},\beta}(\cdot,\cdot)$, upon a call to this oracle for an honest user index $i \in [n_x]$, message inputs $(x_i^{\ell,0}, x_i^{\ell,1})$ (where $x_i^{\ell,b} = \left(x_{i,\mathsf{pub}}^{\ell,b}, x_{i,\mathsf{priv}}^{\ell,b}\right)$ for $b \in \{0,1\}$), the challenger first checks whether the user i was already corrupted or not. That is, if $i \in \mathcal{S}_x$, then it sends nothing, otherwise it samples a ciphertext for input $x_i^{\ell,\beta}$ using key EK_i and sends it to the adversary.

$\mathsf{QKey}^{\mathsf{K},\beta}(\cdot,\cdot)$, upon a call to this oracle for an honest user index $j \in [n_y]$, function inputs $(y_j^{k,0}, y_j^{k,1})$ (where $y_j^{k,b} = \left(y_{j,\mathsf{pub}}^{k,b}, y_{j,\mathsf{priv}}^{k,b}\right)$ for $b \in \{0,1\}$), the challenger first checks whether the user j was already corrupted or not. That is, if $j \in \mathcal{S}_y$, then it sends nothing, otherwise it samples a decryption key for function input $y_j^{k,\beta}$ using key MSK_j and sends it to the adversary. (Here β is the challenge bit chosen at the start of the experiment.)

We let Q_x and Q_y be the number of encryption and key generation queries (respectively) that had non-empty responses. Let $\mathcal{Q}_x = \{(i, (x^{\ell,0}, x^{\ell,1}))\}_{\ell\in[Q_x]}$ be the set of ciphertext queries and $\mathcal{Q}_y = \{(j, (y_j^{k,0}, y_j^{k,1}))\}_{k\in[Q_y]}$ be the set of key queries.

We say that an adversary $\mathcal{A}$ is *admissible* if:

1. For each of the encryption and key challenges, the public components of the two challenges are equal, namely $x^{\ell,0}_{\mathsf{pub}} = x^{\ell,1}_{\mathsf{pub}}$ for all $\ell \in [Q_x]$, and $y^{k,0}_{\mathsf{pub}} = y^{k,1}_{\mathsf{pub}}$ for all $k \in [Q_y]$.
2. For each of the encryption and key challenges, the *private* components of the two challenges are also equal, namely $x^{\ell,0}_{\mathsf{priv}} = x^{\ell,1}_{\mathsf{priv}}$ for all $\ell \in [Q_x]$ whenever $(i, (x^{\ell,0}, x^{\ell,1})) \in \mathcal{Q}_x$ and $i \in \mathcal{S}_x$, and $y^{k,0}_{\mathsf{priv}} = y^{k,1}_{\mathsf{priv}}$ for all $k \in [Q_y]$ whenever $(j, (y^{\ell,0}, y^{\ell,1})) \in \mathcal{Q}_y$ and $j \in \mathcal{S}_y$. That is, the private components must be the same as well if the user index i or j, that the query was made for, was corrupted during the execution.[17]
3. There do not exist two sequences $(\overrightarrow{x}^0, \overrightarrow{y}^0)$ and $(\overrightarrow{x}^1, \overrightarrow{y}^1)$ such that:

$$\mathcal{U}\left(\mathsf{Agg}_x(\{x^0_i\}), \mathsf{Agg}_y(\{y^0_j\})\right) \neq \mathcal{U}\left(\mathsf{Agg}_x(\{x^1_i\}), \mathsf{Agg}_y(\{y^1_j\})\right)$$

and i) for every $i \in [n_x]$, either x^b_i was queried or EK_i was corrupted, and ii) for every $j \in [n_y]$, either y^b_j was queried or MSK_j was corrupted, and iii) at least one of inputs $= x^b_i, = y^b_j$ were queried and indices i, j were not corrupted. (Note that if $i \in [n_x]$ or $j \in [n_y]$ were queried to the QCor oracle, the adversary can generate partial keys or ciphertexts for any value of its choice.)

An MPFE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be IND secure if for any *admissible* PPT adversary $\mathcal{A}$, all length parameters $n_x, n_y \in \mathbb{N}$, and aggregation functions $\mathsf{Agg}_x, \mathsf{Agg}_y$, there exists a negligible function $\mathsf{negl}(\lambda)(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[b' = \beta : \begin{array}{c} \mathsf{K} \leftarrow \mathsf{Setup}(1^\lambda, n_x, n_y, \mathsf{Agg}_x, \mathsf{Agg}_y), \\ \mathsf{K} = (\mathsf{PK}, \{\mathsf{EK}_i\}_i, \{\mathsf{MSK}_j\}_j), \\ \beta \leftarrow \{0,1\}, \\ \beta' \leftarrow \mathcal{A}^{\mathsf{QCor}^{\mathsf{K}}(\cdot), \mathsf{QKey}^{\mathsf{K},\beta}(\cdot), \mathsf{QEnc}^{\mathsf{K},\beta}(\cdot)}(1^\lambda, \mathsf{PK}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Remark B.1 (Weaker notions of security). We say the scheme is selective IND secure if the adversary outputs the challenge message and function pairs at the very beginning of the game, before it makes any queries or receives the PK. One may also consider the semi-honest setting, where the QCor oracle is not provided, or the case of static corruptions where the adversary provides all its corruptions once and for all at the start of the game.

B.1 Dynamic Multi-Party Functional Encryption

In this section, we define the dynamic notion of multi-party functional encryption (MPFE). We consider the fully dynamic setting where the number of key/ciphertext inputs is unspecified during setup time, and the aggregation functions are also specified only during key generation and encryption times. In the dynamic setting, an interactive or centalized setup is not meaningful since the number of parties is itself not known during setup time, hence we restrict ourselves to the local setup mode for simplicity.

Definition B.2 (Dynamic Multi-Party Functional Encryption). Let $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ be the space of ciphertext inputs and $\mathcal{Y} = \mathcal{Y}_{\mathsf{pub}} \times \mathcal{Y}_{\mathsf{priv}}$ be the space of key inputs. Also, let $\mathcal{PK}$ be the space to which each local public key belongs. A dynamic multi-party functional encryption scheme (MPFE) with local setup is defined as a tuple of 5 algorithms/protocols $(\mathsf{Gsetup}, \mathsf{Lsetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ with the following syntax:

$\mathsf{Gsetup}(1^\lambda)$**:** On input the security parameter, the global setup algorithm samples a globally shared set of public parameters PP.

$\mathsf{Lsetup}(\mathsf{PP})$**:** Given input the public parameters, the local setup algorithm outputs a tuple consisting of local public key PK, an encryption key EK, and a master secret key MSK. (Here the local public key is just regarded as a public identifier for the user, and not given as explicit input to other algorithms since it could always be added to the encryption and/or master secret key.)

[17] This condition is an option. When we would like to claim that $\mathsf{EK}_i/\mathsf{MSK}_i$ does not help to decode $\mathsf{CT}_i/\mathsf{SK}_i$, item 2 should be removed.

$\mathsf{Enc}\left(\mathsf{EK}, i, x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}), \mathsf{Agg}_x\right)$**:** Given input an encryption key EK, user index $i \in [n_x]$, an input $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}})$, and an aggregation function $\mathsf{Agg}_x : (\mathcal{PK} \times \mathcal{X})^{n_x} \to \mathcal{X}^*$ (for some $n_x \in \mathbb{N}$), this algorithm outputs a ciphertext CT_i.

$\mathsf{KeyGen}\left(\mathsf{MSK}, j, y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}), \mathsf{Agg}_y\right)$**:** Given input a master secret key MSK, user index $j \in [n_y]$

$\mathsf{Dec}\left((\mathsf{SK}_j)_j, (\mathsf{CT}_i)_i\right)$**:** Given input a sequence of secret keys $(\mathsf{SK}_j)_j$ and a sequence of ciphertexts $(\mathsf{CT}_i)_i$, this algorithm outputs a value z or $\perp$.

Correctness. We say that an MPFE scheme is *correct* if, $\forall (N, n_x, n_y) \in \mathbb{N}^3$, ciphertext inputs $x_i \in \mathcal{X}$ for $i \in [n_x]$, key inputs $y_j \in \mathcal{Y}$ for $j \in [n_y]$, message and function aggregation circuits Agg_x and Agg_y, and indexing functions $\mathrm{index}_x : [n_x] \to [N]$, $\mathrm{index}_y : [n_y] \to [N]$ it holds that:

$$\Pr\left[z = z' : \begin{array}{ll} \mathsf{PP} \leftarrow \mathsf{Gsetup}(1^\lambda) & \\ (\mathsf{PK}_\ell, \mathsf{EK}_\ell, \mathsf{MSK}_\ell) \leftarrow \mathsf{Lsetup}(\mathsf{PP}) & \forall \ell \in [N] \\ \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{EK}_{\mathrm{index}_x(i)}, i, x_i, \mathsf{Agg}_x) & \forall i \in [n_x] \\ \mathsf{SK}_j \leftarrow \mathsf{KeyGen}(\mathsf{MSK}_{\mathrm{index}_y(j)}, j, y_j, \mathsf{Agg}_y) & \forall j \in [n_y] \\ z \leftarrow \mathsf{Dec}\left((\mathsf{SK}_j)_{j \le n_y}, (\mathsf{CT}_i)_{i \le n_x}\right) & \\ z' = \mathcal{U}\left(\mathsf{Agg}_x\left((\mathsf{PK}_{\mathrm{index}_x(i)}, x_i)_i\right), \mathsf{Agg}_y\left((\mathsf{PK}_{\mathrm{index}_y(j)}, y_j)_j\right)\right) & \end{array}\right] = 1.$$

Recall that $\mathcal{U}$ is the universal circuit with appropriate input and output size.

Indistinguishability based security. Here we extend the security experiment for multi-party functional encryption that we provided in Definition B.1 to the dynamic user setting in the local setup mode. Since we are working in the dynamic setting, we need to define the following oracles

$\mathsf{HonestGen}()$, upon a call to this oracle, the challenger samples a fresh tuple of local public key, encryption key, and master key $(\mathsf{PK}, \mathsf{EK}, \mathsf{MSK})$, and stores them in a list $_{\mathsf{setup}}$. It sends PK to the adversary. (Note that if the scheme is a public key scheme, then the challenger sends the encryption key EK to the adversary.)

$\mathsf{QCor}(\cdot, \cdot)$, upon a call to this oracle for an honest user local public key PK and key type $\mathsf{type} \in =$ $\mathsf{enc}, \mathsf{master}$, the challenger first checks whether the list $_{\mathsf{setup}}$ contains a key pair associated with PK. If there is such a key pair $(\mathsf{PK}, \mathsf{EK}, \mathsf{MSK})$, then it sends either the EK or MSK depending on the type queried. Otherwise, it sends nothing.[18]

$\mathsf{QEnc}^\beta(\cdot, \cdot, \cdot, \cdot)$, upon a call to this oracle for an honest user local public key PK, inputs $(x_j^{\ell,0}, x_j^{\ell,1})$ (where $x_j^{\ell,b} = \left(x_{j,\mathsf{pub}}^{\ell,b}, x_{j,\mathsf{priv}}^{\ell,b}\right)$ for $b \in \{0,1\}$), index j, aggregation function $\mathsf{Agg}_{x,j}^\ell$, the challenger first checks whether the list $_{\mathsf{setup}}$ contains a key pair associated with PK. If there is such a key pair $(\mathsf{PK}, \mathsf{EK}, \mathsf{MSK})$, then it samples a ciphertext for input $x_j^{\ell,\beta}$ using key EK and sends it to the adversary. Otherwise, it sends nothing. (Here β is the challenge bit chosen at the start of the experiment.)

$\mathsf{QKey}^\beta(\cdot, \cdot, \cdot, \cdot)$, upon a call to this oracle for an honest user local public key PK, function inputs $(y_j^{k,0}, y_j^{k,1})$ (where $y_j^{k,b} = \left(y_{j,\mathsf{pub}}^{k,b}, y_{j,\mathsf{priv}}^{k,b}\right)$ for $b \in \{0,1\}$), index j, aggregation function $\mathsf{Agg}_{y,j}^k$, the challenger first checks whether the list $_{\mathsf{setup}}$ contains a key pair associated with PK. If there is such a key pair $(\mathsf{PK}, \mathsf{EK}, \mathsf{MSK})$, then it samples a decryption key for function input $y_j^{k,\beta}$ using key MSK and sends it to the adversary. Otherwise, it sends nothing. (Here β is the challenge bit chosen at the start of the experiment.)

We let Q_x and Q_y be the number of encryption and key generation queries (respectively) that had non-empty responses. Let $\mathcal{Q}_x = \{(\mathsf{PK}^\ell, (x_j^{\ell,0}, x_j^{\ell,1}), j, \mathsf{Agg}_{x,j}^\ell)\}_{\ell \in [Q_x]}$ be the set of ciphertext challenge queries and $\mathcal{Q}_y = \{(\mathsf{PK}^k, (y_j^{k,0}, y_j^{k,1}), j, \mathsf{Agg}_{y,j}^k)\}_{k \in [Q_y]}$ be the set of key challenge queries.

We say that an adversary $\mathcal{A}$ is *admissible* if:

[18] As we point out in the static setting, in case EK_i is completely contained in some MSK_i (or vice versa), then making a master secret corruption query for i will also imply that encryption key for i has been corrupted too (and vice versa).

1. For each of the encryption and key challenges, the public components of the two challenges are equal, namely $x^{\ell,0}_{j,\mathsf{pub}} = x^{\ell,1}_{j,\mathsf{pub}}$ for all $\ell \in [Q_x]$, and $y^{k,0}_{j,\mathsf{pub}} = y^{k,1}_{j,\mathsf{pub}}$ for all $k \in [Q_y]$.
2. For each of the encryption and key challenges, the *private* components of the two challenges are also equal, namely $x^{\ell,0}_{j,\mathsf{priv}} = x^{\ell,1}_{j,\mathsf{priv}}$ for all $\ell \in [Q_x]$, and $y^{k,0}_{j,\mathsf{priv}} = y^{k,1}_{j,\mathsf{priv}}$ for all $k \in [Q_y]$ if the encryption key EK^ℓ or the master secret key MSK^k, that the query was made for, was corrupted during the execution (respectively).
3. There do not exist two sequences $((\overrightarrow{\mathsf{PK}}_x, \overrightarrow{x}^0), (\overrightarrow{\mathsf{PK}}_y, \overrightarrow{y}^0)) \neq ((\overrightarrow{\mathsf{PK}}_x, \overrightarrow{x}^1), (\overrightarrow{\mathsf{PK}}_y, \overrightarrow{y}^1))$ and aggregation functions $\mathsf{Agg}_x, \mathsf{Agg}_y$ such that:

$$\mathcal{U}\left(\mathsf{Agg}_x\left((\mathsf{PK}_{x,i}, x^0_i)_i\right), \mathsf{Agg}_y\left((\mathsf{PK}_{y,j}, y^0_j)_j\right)\right) \neq \mathcal{U}\left(\mathsf{Agg}_x\left((\mathsf{PK}_{x,i}, x^1_i)_i\right), \mathsf{Agg}_y\left((\mathsf{PK}_{y,j}, y^1_j)_j\right)\right)$$

and i) x^b_i was queried for aggregation function Agg_x, index i and public key $\mathsf{PK}_{x,i}$, and ii) y^b_j was queried for aggregation function Agg_y, index j and public key $\mathsf{PK}_{y,j}$, and iii) at least one of inputs $= x^b_i, = y^b_j$ were queried and public key $\mathsf{PK}_{x,i}, \mathsf{PK}_{y,j}$ was not corrupted. Note that if some x^b_i or y^b_j was not queried by the adversary, then it can generate partial keys or ciphertexts for any value of its choice by performing a fresh key generation since this is a fully dynamic system, however that samples a fresh public as well.

An MPFE scheme $(\mathsf{Gsetup}, \mathsf{Lsetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is said to be IND secure if for any *admissible* PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)(\cdot)$ such that for all $\lambda \in \mathbb{N}$, the following holds

$$\Pr\left[\mathcal{A}^{\mathsf{HonestGen}(),\ \mathsf{QCor}(),\ \mathsf{QKey}^\beta(),\ \mathsf{QEnc}^\beta()}(1^\lambda) = \beta : \beta \leftarrow \{0,1\}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda).$$

Remark B.2 (Potential variations). The above multi-party function encryption system that we define allows the users to dynamically join the system in the permissionless model, where each incoming user only needs to know the public parameters and not interact with any authority. A slightly weaker setting could be a permissioned model in which users can still dynamically join the system but they need to contact the global authority (which sampled the public parameters) either for some identification tokens or its encryption and master secret key pair in order to prevent totally unrestricted computation which happens in the permissionless model.

Also, we want to point out that in our current framework we let the users select the aggregation functions during individual functional key and ciphertext generation to allow for more flexibility. This could be relaxed even further by letting the aggregation functions be either be described in a uniform computation model, or using an ensemble of non-uniform functions. Also, one could instead restrict the flexibility in aggregation by asking each user to choose their aggregation functions at setup time. Such flexibilities will be important in capturing the notion of DDFE described in Definition 6.1.

B.2 Capturing our primitives in the MPFE framework

They also proposed a dynamic variant of MPFE, which is presented in Appendix B.1. In this paper, we use following variants of FE subsumed by MPFE or dynamic MPFE. Formal definitions of these are found in Section 2.3 or respective sections.

Attribute-Based Encryption. Attribute-based encryption (ABE) for predicate $\mathsf{P} : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ is captured by MPFE as follows: $(n_x, n_y) = (1,1)$, $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = (x', M)$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (y', \perp)$. $\mathsf{Agg}_x(x) = x$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}(x, f) = M$ if $\mathsf{P}(x', y') = 1$ and $\mathcal{U}(x, f) = \perp$ otherwise.

Secret-Key Functional Encryption. Secret-key functional encryption (SK-FE) for function class $\mathcal{F}$ is captured by MPFE as follows: $(n_x, n_y) = (1,1)$, $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = (x_1, x_2)$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (f_1, f_2) = f \in \mathcal{F}$. $\mathsf{Agg}_x(x) = x$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}(x, f) = f(x)$.

Multi-Input Functional Encryption. Multi-input functional encryption (MIFE) for function class $\mathcal{F}$ is captured by MPFE as follows: $(n_x, n_y) = (n, 1)$, $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}}) = (x_{i,1}, x_{i,2})$, $y =$

$(y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (f_1, f_2) = f \in \mathcal{F}$. $\mathsf{Agg}_x(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}((x_1, \ldots, x_n), f) = f(x_1, \ldots, x_n)$.

Multi-Client Functional Encryption. Multi-client functional encryption (MCFE) for function class $\mathcal{F}$ is captured by MPFE as follows: $(n_x, n_y) = (n, 1)$, $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}}) = ((x_{i,1}, L_i), x_{i,2})$, $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (f_1, f_2) = f \in \mathcal{F}$. $\mathsf{Agg}_x(x_1, \ldots, x_n) = (x_1, \ldots, x_n)$, and $\mathsf{Agg}_y(y)$ outputs f such that $\mathcal{U}((x_1, \ldots, x_n), f) = f((x_{1,1}, x_{1,2}), \ldots, (x_{n,1}, x_{n,2}))$ if and only if $L_1 = \cdots = L_n$.

Dynamic Decentralized Functional Encryption. Dynamic decentralized functional encryption (DDFE) for function F is captured by dynamic MPFE (Appendix B.1) as follows: $x_i = (x_{i,\mathsf{pub}}, x_{i,\mathsf{priv}}) = (m_{i,1}, m_{i,2}) = m_i$, $y_i = (y_{i,\mathsf{pub}}, y_{i,\mathsf{priv}}) = (k_{i,1}, k_{i,2}) = k_i$. Agg_x is an identity function, and $\mathsf{Agg}_y(\{\mathsf{PK}_i, k_i\}_{i \in \mathcal{U}_K})$ outputs f such that $\mathcal{U}(\{\mathsf{PK}_i, m_i\}_{i \in \mathcal{U}_M}, f) = F(\{\mathsf{PK}_i, m_i\}_{i \in \mathcal{U}_M}, \{\mathsf{PK}_i, k_i\}_{i \in \mathcal{U}_K})$. Note that we assume that aggregate functions here can be described in non-uniform computation model such as Turing machines as in Remark B.2.

Attribute-Based FE for Attribute-Weighted Sums with Inner Product We can capture Attribute-Based FE for AWS with Inner Product in the context of MPFE as follows. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The setup algorithm is run in the $\mathsf{central}$ mode, and $n_x = n_y = 1$. A message is defined as $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = ((\mathbf{y}, \{\mathbf{x}_j\}_{j \in [N]}), (\{\mathbf{z}_j\}_{j \in [N]}, [\mathbf{p}]_1))$ where $\mathbf{y}, \mathbf{x}_j, \mathbf{z}_j, \mathbf{p}$ are all vectors in $\mathbb{Z}_p$ while a function is defined as $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = ((g, h), [\mathbf{q}]_2)$ where g, h are ABPs, and $\mathbf{q}$ is a vector in $\mathbb{Z}_p$. Agg_x is an identity function, and Agg_y outputs a function $f_{g,h,[\mathbf{q}]_2}$ that outputs $[\sum_{j \in [N]} \langle h(\mathbf{x}_i), \mathbf{z}_i \rangle + \langle \mathbf{p}, \mathbf{q} \rangle]_T$ if and only if $g(\mathbf{y}) = 0$ on input $x = ((\mathbf{y}, \{\mathbf{x}_j\}_{j \in [N]}), (\{\mathbf{z}_j\}_{j \in [N]}, [\mathbf{p}]_1))$.

Attribute-Based MIFE for Attribute-Weighted Sums We can capture Attribute-Based MIFE for AWS in the context of MPFE as follows. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The setup algorithm is run in the $\mathsf{central}$ mode, and $n_x = n, n_y = 1$ for some $n \in \mathbb{N}$. A message is defined as $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = ((\mathbf{y}, \{\mathbf{x}_j\}_{j \in [N]}), \{\mathbf{z}_j\}_{j \in [N]})$ where $\mathbf{y}, \mathbf{x}_j, \mathbf{z}_j$ are all vectors in $\mathbb{Z}_p$ while a function is defined as $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = (\{g_i, h_i\}_{i \in [n]}, \bot)$ where g_i, h_i are ABPs. Agg_x is an identity function, and Agg_y outputs a function $f_{\{g_i, h_i\}_{i \in [n]}}$ that outputs $[\sum_{i \in [n]} \sum_{j \in [N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle]_T$ if and only if $g_i(\mathbf{y}_i) = 0$ for all $i \in [n]$ on input $\{x_i\}_{i \in [n]} = \{(\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_{j \in [N_i]}), \{\mathbf{z}_{i,j}\}_{j \in [N_i]}\}_{i \in [n]}$.

Dynamic Decentralized Functional Encryption for AWS We can capture DDFE for AWS in the context of dynamic MPFE as follows. Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. The setup algorithm is run in the local mode, since it works in a dynamic manner. A message is defined as $x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) = ((\{\mathbf{x}_j\}_{j \in [N]}, \mathcal{U}_M, L_M), \{\mathbf{z}_j\}_{j \in [N]})$ where $\mathbf{x}_j, \mathbf{z}_j$ are vectors in $\mathbb{Z}_p$, $\mathcal{U}_M$ is a set of IDs, and L_M is a label while a function is defined as $y = (y_{\mathsf{pub}}, y_{\mathsf{priv}}) = ((\{f_i\}_{i \in \mathcal{U}_K}, \mathcal{U}_K), \bot)$ where f is an ABP. Agg_x checks if the public inputs $(\mathcal{U}_M, L_M)$ match for all parties and that all the ciphertexts are provided for the set $\mathcal{U}_M$. If so, it outputs $(\{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{i \in \mathcal{U}_M, j \in [N_i]}, \mathcal{U}_M)$. Agg_y checks $(\{f_i\}_{i \in \mathcal{U}_K}, \mathcal{U}_K)$ match for all parties and that all the ciphertexts are provided for the set $\mathcal{U}_K$. If so, it outputs a function f'_y that outputs $[\sum_{i \in \mathcal{U}_K} \sum_{j \in [N_i]} \langle f_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle]_T$ if and only if $\mathcal{U}_M = \mathcal{U}_K$ on input $(\mathcal{U}_M, \{\mathbf{x}_{i,j}, \mathbf{z}_{i,j}\}_{i \in \mathcal{U}_M, j \in [N_i]})$.