Dynamic Decentralized Functional Encryption: Generic Constructions with Strong Security

Ky Nguyen¹[®], David Pointcheval^{1,2}[®], and Robert Schädlich¹[®]

 1 DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France 2 Cosmian, Paris, France

Abstract. Dynamic Decentralized Functional Encryption (DDFE) is a generalization of Functional Encryption which allows multiple users to join the system dynamically without interaction and without relying on a trusted third party. Users can independently encrypt their inputs for a joint evaluation under functions embedded in functional decryption keys; and they keep control on these functions as they all have to contribute to the generation of the functional keys.

In this work, we present new generic compilers which, when instantiated with existing schemes from the literature, improve over the state-of-the-art in terms of security, computational assumptions and functionality. Specifically, we obtain the first adaptively secure DDFE schemes for inner products in both the standard and the stronger function-hiding setting which guarantees privacy not only for messages but also for the evaluated functions. Furthermore, we present the first DDFE for inner products whose security can be proven under the LWE assumption in the standard model. Finally, we give the first construction of a DDFE for the attribute-weighted sums functionality with attribute-based access control (with some limitations). All prior constructions guarantee only selective security, rely on group-based assumptions on pairings, and cannot provide access control.

Table of Contents

1	Intr	oduction	3
	1.1	Research Questions	5
	1.2	Our Contributions	6
2	Pre	liminaries	7
	2.1	Notational Conventions	7
	2.2	Pairing Groups and Hardness Assumptions	8
	2.3	Arithmetic Branching Programs	9
	2.4	Dynamic Decentralized Functional Encryption	9
	2.5	Decentralized Multi-Client Functional Encryption	12
	2.6	Attribute-Based and Identity-Based Encryption	14
	2.7	Lockable Obfuscation	16
	2.8	Pseudorandom Functions (PRF)	16
	2.9	Non-Interactive Key Exchange (NIKE)	17
3	Tecl	hnical Overview	18
	3.1	From DMCFE to DDFE	18
	3.2	Achieving Security Against Any Queries	21
4	From	m DMCFE to DDFE	24
5	Secu	urity Against Any Queries	29
	5.1	Attribute-Based All-or-Nothing Encapsulation	29
	5.2	DDFE for AB-AWS Secure Against Any Queries	34
А	Sup	porting Materials – Section 4	42
	A.1	Generic Upgrade of the Security Model in [NPS24]	42
	A.2	Instantiation of Construction 25 with a DMCFE for AB-AWS	44
	A.3	Instantiation of Construction 25 with the DMCFE of [CDG ⁺ 18a]	48
	A.4	Instantiation of Construction 25 with a Variant of [LT19]	49
	A.5	Instantiation of Construction 25 with the DMCFE of [NPS24]	56

1 Introduction

Functional Encryption. Public-Key Encryption (PKE) is one of the fundamental objects of studies in modern cryptography. Starting from the initial interest of *all-or-nothing* security - either a decryption key reveals the whole original message or the ciphertext is sementically secure - the past decades have witnessed new interests in a more fine-grained control over the information that can be obtained at decryption time. Boneh, Sahai and Waters [SW05, BSW11] introduced the concept of *Functional Encryption* (FE) in which a decryption key is *functional* with respect to some function f and decrypting a ciphertext under this functional key reveals only f(x) for a plaintext x. The essential lies in limiting the information leakage about x to not more than what is revealed by f(x). Previous advanced notions of PKE, such as *Identity-Based Encryption* [Sha84, Coc01, BF01] or Attribute-Based Encryption [SW05, GPSW06, OSW07, ALdP11, OT12], are enclosed in the umbrella of FE. More importantly, FE for general function classes shows versatile connections to other important cryptographic primitives [ABF⁺13, AJ15, BV15, Dat20, BS23] as well as motivates the study of new notions [GVW15]. Zooming in more concrete classes, notably constructing FE schemes to evaluate *inner products* between a functional vector in the key and a data vector in the ciphertext. aka IPFE, has received a great deal of attention since the seminal work of Abdalla et al. [ABDP15]. Thus emerge further improved IPFE constructions [ALS16, BBL17, CLT18] and novel candidates for more complex concrete evaluations, such as for quadratic functions [BCFG17, Gay20, AS17, Lin17] or for attribute-weighted sums [AGW20, ATY23].

Properties of Functional Decryption Keys. One important flexibility of FE is that one can enrich further properties of the function class that are reflected via the functional keys. In this paper we focus on two aspects: *controlling the usage of functional keys* and *hiding the function*.

Fine-Grained Control over Functional Decryption Keys. The notion of FE tackles the leakage of information about the plaintext x in terms of the function evaluation f(x). It remains another classical question, which concerns the functional decryption keys themselves: without further restrictions, once a key is obtained, it can be used forever. Given the inherent information that can be gathered from each key, this poses a serious threat when we look at concrete function classes. For instance, in IPFE schemes, when the obtained keys correspond to a basis of the inner product vector space, the adversary can easily recover the whole plaintext vector - by decrypting to get the inner products and solving a linear system. The problem of controlling decryption keys is the subject of extensive studies in the literature of broadcast encyrption, revocation systems, and more generally, of ABE itself, e.g. see [Wee21, Wee22, FWW23] and related works. For practical interests³, Abdalla et al. [ACGU20] started the line of works on integrating a mechanism of *attribute-based access control* into FE for inner products. Follow-up works include [NPP22, NPP24] for inner products and [ATY23] for the class of attribute-weighted sums. It is important to note that the foregoing access-control over decryption keys in FE is part of an enhancement to the function class, as argued in [NPP22] and later made clear in [ATY23, NPP24]. Therefore, for example, moving from IPFE to IPFE with access control incurs dealing with a richer function class and is more challenging.

<u>Function Privacy</u>. Originally, FE is an advanced PKE notion and it provides security regarding the plaintext x vis-a-vis the ciphertexts and the functional decryption keys. The security model of FE per se does not require the functional keys to hide the function they decrypt. From a practical standpoint, however, the function itself can contain confidential data and it is of interests to require more confidentiality with respect to the functional decryption keys. Specific examples include models of machine/deep learning whose parameters constitute the function to be evaluated, and these

³ The problem of controlling keys becomes easy if one is willing to resort to FE schemes for general function class, at the cost of prohibitively poor efficiency.

parameters are often the result of expensive training processes, which are kept secret to ensure the business model of the service provider. It is then natural to require that the function itself is hidden from the adversary, which is known as the *function-hiding* (FH) property. Apart from the practical advantages, FH-FE schemes also turn out to be an important theoretical object. From a FH-IPFE, various works [Lin17, Gay20] exploit different approaches to achieve FE schemes for quadratic functions. Recent works [AGT21a, AGT22] go even further, still based on FH-IPFE-compatible approaches, to successfully realize quadractic evaluation in the *multi-input* setting. Sticking to the class of inner products, the property of function-hiding itself inspires recent progress in the *multi-user* setting, *e.g.* see [SV23, NPS24, Ngu24]. As a consequence, we believe that the function-hiding property is a desirable aspect of FE schemes that deserves to be studied in its own right, especially when considering the practical applications and the technical building blocks for more advanced constructions. We will revise the multi-user/inputs setting below.

Extensions of FE in the Multi-User Setting. The evaluational nature of FE gives rise to immediate extensions to the *multi-user* setting, in which the function in the functional key will evaluate over multiple inputs that can come from different sources. Hence, soon after the seminal work of Boneh *et al.* [SW05, BSW11] on FE, the study of *Multi-Client Functional Encryption* (MCFE) and *Multi-Input Functional Encryption* (MIFE) was initiated by Goldwasser *et al.* [GGG⁺¹⁴, GKL⁺¹³]. Since their introduction, a long line of works has been devoted to the study of MIFE/MCFE, notably for the concrete function class of inner products [DOT18, CDG^{+18a}, CDG^{+18b}, ACF⁺¹⁸, ABKW19, ABG19, LT19, CDSG⁺²⁰, ACGU20, NPP22] and more, *e.g.* for quadratic functions [AGT21a, dPP22, AGT22] or for attribute-weighted sums [ATY23].

(Decentralized) Multi-Client Functional Encryption. As a recapitulation from the foundational works of [GGG⁺14, GKL⁺13, CDG⁺18a] about MIFE/MCFE, the setting of MCFE captures the scenario where multiple users independently encrypt their data using their private encryption keys under some message tag. The functional key is generated by a trusted third party that can be used to evaluate the function over the multiple ciphertexts, only if all ciphertexts share the same message tag tag. Taking a step further, the work by Chotard *et al.* $[CDG^{+}18a]$ questions the need of a central authority for distributing functional keys in MCFE and proposes the Decentralized Multi-Client Functional Encryption (DMCFE). In DMCFE, the interaction between users is done during the setup of the system so as to generate a secret key SK_i that is used for both encryption and functional key generation, for each user i. Thanks to the ensemble of the secret keys SK_i , the functional key is now generated without the need of a central authority: given a secret key SK_i , a user can independently contribute to the generation of a functional key DK_f for a function f, and each contribution is associated with a key tag tag-f for later joint combination, where tag-f can contain the description of fitself. This liberation of DMCFE from the necessity of a central authority is a significant improvement in terms of trust model, and unsurprisingly after [CDG⁺18a] many follow-up works dive into different angles, ranging from constructions and applications [LT19, ABKW19, ABG19, QLH⁺24] to security enhancements [NPP23, NPS24]. In terms of the supported functions, all aforementioned works focus on the class of inner products, while some progress for more expressive classes to compute attribute-weighted sums is only achieved very recently in [ATY23, Ngu24]. More interestingly, the two works [ATY23, Ngu24] focus on a more general version of DMCFE, which is coined Dynamic Decentralized Functional Encryption (DDFE) and in the end gives implicit constructions of DMCFE. It is indeed this utmost general notion of DDFE with which the current work is concerned. We will come back to the notion of DDFE in the following, after addressing a technical subtlety of the message/key tags up to the current notion of DMCFE.

<u>Repetitions under One Message/Key Tag.</u> It is made clear in the above discussion that only ciphertexts of the same message tag tag and functional keys of the same key tag tag-f can be

combined in the notion of DMCFE. A question that immediately comes to mind is the role of these message/key tags in the security model of DMCFE. The initial security model of DMCFE in [CDG⁺18a] and subsequent work [LT19] ignores different queries for the same pair (i, tag) or (i, tag-f) after the first query. One justification that is given in [CDG⁺18a] is that it is up to the user's responsibility not to use the same mesage and key tag twice, neither for encryption nor for key-generation, respectively. Nonetheless, we believe that proving security under a repeated usage of tags is still important. First of all, from a practical point of view, there can exist a scenario where a user mistakenly or maliciously uses the same tag twice. Secondly, from a theoretical point of view, when *repetitions* are allowed for the message tag in ciphertexts, *i.e.* security still holds even when the adversary can obtain multiple ciphertexts for the same (i, tag), it is studied in preceding works [ABKW19, ABG19] and recently confirmed in [ATY23, NPP24] that the security model of (D)MCFE encircle that of MIFE, where in the former we can fix a public message tag for all ciphertexts and arrive at the latter in which there is no message tag whatsoever. As a consequence, our aim for the security of DMCFE and the like will allow repetitions on both message/key tags.

Dynamic Decentralized Functional Encryption. As explained in the preceding paragraphs, in DM-CFE some interaction is required during the setup phase of the system, which implies that the number of users is fixed in advance. Embarking on the resolution of this rigidity, Chotard *et al.* $[CDSG^+20]$ generalized DMCFE and defined the notion of Dynamic Decentralized Functional Encryption (DDFE) where users can join at various stages during the lifetime of a system. All decentralized features of DMCFE are maintained in DDFE. The striking difference is that in DDFE there is only a non-interactive global setup, whereas each user when joining can run a local setup algorithm to generate their own secret key SK_i using some public parameters set by that *global* setup algorithm. At any time, any set of users \mathcal{U}_M can independently encrypt their individual data to contribute to a list of ciphertexts $(CT_i)_{i \in \mathcal{U}_M}$ under some message tag tag. Similarly, a set of users \mathcal{U}_K can independently contribute to a list of functional keys $(DK_i)_{i \in \mathcal{U}_K}$ under some key tag tag-f. We recall the usage of tags here is similar to that in (D)MCFE: the ciphertexts and the functional keys can be combined only if they have the same message and key tags, respectively. A DDFE scheme allows jointly decrypting a list of ciphertexts $(CT_i)_{i \in \mathcal{U}_M}$ using a list of functional keys $(DK_i)_{i \in \mathcal{U}_K}$, without resorting to any central authority. Construction-wise, the work of $[CDSG^+20]$ gives a DDFE for the class of inner products (IP-DDFE) at the core of which is the MCFE from $[CDG^{+}18a]$. The work of [CDSG⁺20] is then followed by [AGT21b] that revisits and improves by first constructing a FH-MCFE for inner products and then lifting it to a FH-DDFE for inner products. Moving away from inner products, [ATY23] presents the first DDFE for the more general class of attributeweighted sums (AWS). All constructions attain only selective security under static corruption in the ROM, *i.e.* the adversary makes all encryption, key-generation and corruption queries up front in one shot. Very recently, [Ngu24] leverages the state-of-the-art from [AGT21b, ATY23] to give the first FH-DDFE for inner products and DDFE for AWS without ROM. All mentioned works allow repetitions for message tags. Regarding the security against repetitions on key tags. [AGT21b] and the FH-IP-DDFE of [Ngu24] explicitly exclude them from their security model, whereas [CDSG⁺20, ATY23] and the DDFE for AWS of [Ngu24] consider a simplified functionality that does not consider key generation with respect to tags, thus there is no notion of repetition for key tags.

1.1 Research Questions

The above expository discussion leads us to various research questions. This paper focuses on DDFE for inner products or for attribute-weighted sums, with or without access-control, with or without

function-hiding. The following questions are not necessarily distinct, but we dissect them on different facets for clarity. Needless to say, resolving multiple of them at the same time is notoriously hard:

- 1. All cited DDFE schemes [CDSG⁺20, AGT21b, ATY23] attain only *selective* security under *static* corruption. *How far can we push for adaptive security of DDFE?* Note: the concurrent and independent work [Ngu24] also achieves adaptively secure FH-IP-DDFE under static corruption.
- 2. Regarding the attribute-based access control over functional keys, existing works can go as far as MCFE or MIFE, for inner products in [ACGU20, NPP22, NPP24] and for attribute-weighted sums in [ATY23]. How further can we integrate access control into the multi-user setting, e.g. for DDFE?
- 3. All cited DDFE schemes [CDSG⁺20, AGT21b, ATY23, Ngu24] rely on group-based assumptions and do not provide post-quantum security. The only multi-user FE scheme in the post-quantum regime comes from the DMCFE of [LT19] for inner products and relies on the *Learning with Error* (LWE) assumption. *How far can we push for post-quantum security for DDFE*?
- 4. The security against repetitions on key tags is either excluded in the FH-IP-DDFE from [AGT21b, Ngu24], or not explicitly considered in [CDSG⁺20] for IP-DDFE and in [ATY23, Ngu24] for AWS-DDFE. How can we achieve security against repetitions for both encryption and key-generation queries in the FH-DDFE and/or DDFE with access control setting?

1.2 Our Contributions

In this paper, we present several candidates of DDFE for the classes of inner products and attributeweighted sums that strictly improve on various aspects of security compared with [CDSG⁺20, AGT21b, ATY23, Ngu24]. At the center of our constructions are two generic transformations that enjoy preferable properties:

- 1. Compiler 1 From DMCFE to DDFE. Inspired by the (non-generic) lifting result from FH-IP-MCFE to FH-IP-DDFE of [AGT21b], we present a generic conversion from DMCFE to DDFE that works both in the standard and function-hiding setting and for arbitrary functionalities, while preserving the security properties of the DMCFE. For the compiler to be applicable, we require some natural structural properties that are satisfied by many DMCFE schemes in the literature. Details are given in Section 4.
- 2. Compiler 2 From Legitimate-Query to Any-Query Security. For function classes with access control, we present a generic transformation to remove the so-called *legitimate-query* constraint. Intuitively, this constraint requires that for each message/key tag combination, the adversary either queries for ciphertexts and keys that permit an honest decryption (*i.e.* the attributes of the ciphertexts are authorized with respect to the key's policy), or no queries are submitted at all. To achieve a polynomial runtime of the encryption and decryption procedure, we must limit the number of function tags to some a priori fixed polynomial and the number of users who can jointly evaluate a function to a constant. Details are given in Section 5.

Below and in Table 1 are presented a summary of our final DDFE instantiations and a comparison with existing works. All DDFE go through the first compiler from DMCFE to DDFE. For the case of AB-AWS, we additionally apply the second compiler to remove the legitimate-query constraint.

1. Concrete Instantiations - IP-DDFE. With respect to the inner-product functionality, we show how to instantiate our compiler with the DMCFE schemes of [CDG⁺18a, LT19]. In this way, we obtain the first adaptively secure IP-DDFEs, with tag repetition under the SXDH assumption in the ROM, and without tag repetition under the LWE assumption in the standard model. This provides an affirmative answer to questions 1, 3 and 4 in the case of IP-DDFE.

	Functionality	FH	Oracle Queries			
Scheme			ØEnc	ØKeyGen	Assumptions	ROM
$[CDSG^+20]$	inner products	×	$sel, \underline{w/-rep}$	sel, 🗔	SXDH	1
[AGT21b]	inner products	⊻	$sel, \underline{w/-rep}$	sel, w/o-rep	SXDH	1
[ATY23]	AWS	×	sel, w/-rep	$ $ sel, \square	MDDH	1
[Ngu24]	inner products	∠	adap, w/-rep	adap, w/o-rep	SXDH	×
[Ngu24]	AWS	×	$sel, \underline{w/-rep}$	sel, 🗔	SXDH	<u>×</u>
Sec. $4 + [CDG^+18a]$	inner products	x	adap, w/-rep	adap, w/-rep	SXDH	1
Sec. $4 + [LT19]$	inner products	×	$\underline{adap}, w/o-rep$	adap, w/o-rep	LWE	×
Sec. $4 + [NPS24]$	inner products	∠	adap, w/-rep	$\underline{adap}, bnd-rep^{\dagger}$	SXDH	1
Sec. $4 + 5 + A.2$	AB-AWS [‡]	×	sel, w/-rep	adap, w/-rep	LWE, SXDH	1

The work considers a simpler functionality without function tags, thus there is no notion of function tag repetition.

[†] The number of repetitions at one slot for any key tag is fixed polynomially bounded.

 ‡ The construction suffers from two limitations: the number of supported function tags is polynomially

bounded and the maximum number of users who can jointly evaluate a function is a constant.

Table 1: We compare our obtained DDFE with existing works, in terms of whether the scheme provides standard security (λ) or the stronger function-hiding security (λ) (FH), whether the encryption oracle ($\mathcal{O}Enc$) and key-generation oracle ($\mathcal{O}KeyGen$) can be queried adaptively and with repetitions (**Oracle Queries**), which assumptions are used for the security proof (**Assumptions**), and whether the security is proven in the ROM (λ) or not (λ) (**ROM**). The shorthands (sel, adap) denote selective or adaptive oracle queries. The shorthands (w/-rep, w/o-rep) indicate whether the adversary can demand repetitive queries to the same slot and tag or not. All schemes consider only static corruptions, and all schemes from group-based assumptions use pairings. Preferred properties are <u>underlined</u>.

- 2. Concrete Instantiation FH-IP-DDFE. In the function-hiding setting of the inner-product functionality, we instantiate our compiler with the FH-DMCFE scheme of [NPS24]. This gives the first adaptively secure FH-IP-DDFE, with full repetitions on message tags and an a priori polynomially bounded number of repetitions on key tags under the SXDH assumption in the ROM. This provides an affirmative answer to question 1 and partially resolves question 4 in the case of FH-IP-DDFE.
- 3. Concrete Instantiation AB-AWS-DDFE. Beyond inner products towards AB-AWS, we instantiate our compilers with a new DMCFE for AWS with access control that is also constructed in this work. We achieve semi-adaptive security (*i.e.* the encryption oracle cannot be called anymore after the first key generation query) under the SXDH and LWE assumptions in the ROM, thereby resolving question 2 and partially question 1 in the case of DDFE for AB-AWS.

2 Preliminaries

2.1 Notational Conventions

Let $\lambda \in \mathbb{N}$ be the security parameter. Except in the definitions, we will suppress λ in subscripts for brevity. A nonnegative function $\varepsilon \colon \mathbb{N} \to \mathbb{R}$ is negligible if $\varepsilon(\lambda) = O(\lambda^{-n})$ for all $n \in \mathbb{N}$. An algorithm is said to be *efficient* if it runs in probabilistic polynomial time (PPT) in the security parameter.

To avoid confusion, we always write vectors \mathbf{v} and matrices \mathbf{A} in boldface and use uppercase letters for the latter. Scalars *s* are written in italics. Unless otherwise stated, all vectors \mathbf{v} are viewed as column vectors. The corresponding row vector is denoted by \mathbf{v}^{\top} . Security Experiments and Distributions. Let \mathbf{Exp} be an interactive *experiment* that interacts with an algorithm \mathcal{A} (called the *adversary*), depends on the security parameter λ and has binary outcome. We also refer to such objects as *games* or *hybrids*. We let " $\mathbf{Exp}_{\mathcal{A}}(1^{\lambda}) \to 1$ " denote the event that the outcome of running \mathbf{Exp} with \mathcal{A} on security parameter λ is 1. For two experiments \mathbf{Exp}^{0} and \mathbf{Exp}^{1} , we define the distinguishing advantage of \mathcal{A} against the tuple ($\mathbf{Exp}^{0}, \mathbf{Exp}^{1}$) as

$$\mathbf{Adv}_{\mathbf{Exp}^{0},\mathbf{Exp}^{1},\mathcal{A}}(\lambda) \coloneqq \left| \Pr\left[\mathbf{Exp}^{1}_{\mathcal{A}}(1^{\lambda}) \to 1\right] - \Pr\left[\mathbf{Exp}^{0}_{\mathcal{A}}(1^{\lambda}) \to 1\right] \right|$$

We write $\mathbf{Exp}^0 \approx_c \mathbf{Exp}^1$ if the experiments are *computationally indistinguishable*, *i.e.* their distinguishing advantage is negligible for all efficient adversaries \mathcal{A} . We write $\mathbf{Exp}^0 \approx_s \mathbf{Exp}^1$ if the experiments are *statistically indistinguishable*, *i.e.* their distinguishing advantage is negligible for all (even unbounded) adversaries. We write $\mathbf{Exp}^0 \equiv \mathbf{Exp}^1$ if the experiments are *identically distributed*, *i.e.* their distinguishing advantage is 0 for all (even unbounded) adversaries. By default, the term *indistinguishable* refers to computational indistinguishability.

More general, the same notations can be used for sequences of distributions. Let $D^0 = \{D^0_\lambda\}_{\lambda \in \mathbb{N}}$ and $D^1 = \{D^1_\lambda\}_{\lambda \in \mathbb{N}}$ be two sequences of distributions. For $b \in \{0, 1\}$, we define $\operatorname{Exp}_{\mathcal{A}}^b(1^\lambda)$ as follows: sample $x \stackrel{s}{\leftarrow} D^b_\lambda$, run $\mathcal{A}(1^\lambda, x)$ and use the output of \mathcal{A} as the outcome of the experiment. Then we write $D^0 \approx_c D^1$ (resp. $D^0 \approx_s D^1$, $D^0 \equiv D^1$) if $\operatorname{Exp}_{\mathcal{A}}^0 \approx_c \operatorname{Exp}_{\mathcal{A}}^1$ (resp. $\operatorname{Exp}_{\mathcal{A}}^0 \approx_s \operatorname{Exp}_{\mathcal{A}}^1$, $\operatorname{Exp}_{\mathcal{A}}^0 \equiv \operatorname{Exp}_{\mathcal{A}}^1$).

Sets and Indexing. We denote by \mathbb{Z} and \mathbb{N} the sets of integers and natural numbers (positive integers). For integers m and n, we write [m; n] to denote the set $\{z \in \mathbb{Z} : m \leq z \leq n\}$ and let [n] := [1; n]. For a prime number p, \mathbb{Z}_p denotes the finite field of integers modulo p. For a finite set S, we let 2^S denote the power set of S. To index a vector or the columns of a matrix, we write $\mathbf{v}[i]$ and $\mathbf{A}[j]$. In contrast, objects of some collection that is not regarded as a vector or matrix are indexed using subscripts (or superscripts in some cases). For instance, \mathbf{v}_i represents a vector, not a component of some vector. If i runs through some index set [n], it means that there are n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$. If the n objects are scalars (or not explicitly vectors), we will write v_1, \ldots, v_n instead.

2.2 Pairing Groups and Hardness Assumptions

Pairing Groups. Our constructions use a sequence of pairing groups

$$\mathbb{G} = \{\mathbb{G}_{\lambda} = (\mathbb{G}_{\lambda,1}, \mathbb{G}_{\lambda,2}, \mathbb{G}_{\lambda,t}, g_{\lambda,1}, g_{\lambda,2}, g_{\lambda,t}, e_{\lambda}, p_{\lambda})\}_{\lambda \in \mathbb{N}}$$

where $\mathbb{G}_{\lambda,1}$ (resp. $\mathbb{G}_{\lambda,2}$, $\mathbb{G}_{\lambda,t}$) is a cyclic group of order p_{λ} generated by $g_{\lambda,1}$ (resp. $g_{\lambda,2}$, $g_{\lambda,t}$), and e_{λ} : $\mathbb{G}_{\lambda,1} \times \mathbb{G}_{\lambda,2} \to \mathbb{G}_{\lambda,t}$ is the pairing operation satisfying $e_{\lambda}(g_{\lambda,1}^{a}, g_{\lambda,2}^{b}) = g_{\lambda,t}^{ab}$ for all integers a, b. The group operations and the pairing map are required to be efficiently computable.

Following the implicit notation in $[\mathbf{EHK}^+13]$, we write $[\![a]\!]_i$ to denote $g^a_{\lambda,i}$ for $i \in \{1, 2, t\}$. This notation extends component-wise to matrices and vectors having entries in \mathbb{Z}_p . Equipped with these notations, group operations are written additively and the pairing operation multiplicatively, *e.g.* $[\![\mathbf{A}]\!]_1 - \mathbf{B}[\![\mathbf{C}]\!]_1 \mathbf{D} = [\![\mathbf{A} - \mathbf{BCD}]\!]_1$ and $[\![\mathbf{A}]\!]_1[\![\mathbf{B}]\!]_2 = [\![\mathbf{AB}]\!]_t$.

Computational Assumptions. We state the assumptions needed for our constructions. Let $\{\mathbb{G}_{\lambda} = (\mathbb{G}_{\lambda,1}, \mathbb{G}_{\lambda,2}, \mathbb{G}_{\lambda,t}, g_{\lambda,1}, g_{\lambda,2}, g_{\lambda,t}, e_{\lambda}, p_{\lambda})\}_{\lambda \in \mathbb{N}}$ be a sequence of pairing groups.

Definition 1 (Decisional Diffie-Hellman Assumption (DDH)). Let $i \in \{1, 2, t\}$. The DDH assumption holds in $\{\mathbb{G}_{\lambda,i}\}_{\lambda \in \mathbb{N}}$ if $\{[\![a, b, ab]\!]_i\}_{\lambda \in \mathbb{N}} \approx_c \{[\![a, b, ab + c]\!]_i\}_{\lambda \in \mathbb{N}}$ for $a, b, c \stackrel{\text{\tiny \$}}{=} \mathbb{Z}_{p_{\lambda}}$.

Definition 2 (Symmetric eXternal Diffie-Hellman Assumption (SXDH)). The SXDH assumption holds in $\{\mathbb{G}_{\lambda}\}_{\lambda \in \mathbb{N}}$ if the DDH assumption holds in both $\{\mathbb{G}_{\lambda,1}\}_{\lambda \in \mathbb{N}}$ and $\{\mathbb{G}_{\lambda,2}\}_{\lambda \in \mathbb{N}}$.

2.3 Arithmetic Branching Programs

We recall the definition of arithmetic branching programs.

Definition 3 (Arithmetic Branching Program (ABP)). An arithmetic branching program $f: \mathbb{Z}_q^{n_0} \to \mathbb{Z}_q$ is defined by a tuple $(V, E, s, t, q, n_0, \sigma)$ consisting of a directed acyclic graph (V, E) with two distinguished vertices $s, t \in V$, a prime q, an arity n_0 and a labelling function $\sigma: E \to \mathcal{F}^{\mathsf{aff}}$, where $\mathcal{F}^{\mathsf{aff}}$ contains all affine functions $g: \mathbb{Z}_q^{n_0} \to \mathbb{Z}_q$. Let P be the set of all paths from s to t. The output of the ABP on input $\mathbf{x} \in \mathbb{Z}_q^{n_0}$ is defined as $f(\mathbf{x}) = \sum_{p \in P} \prod_{e \in p} \sigma(e)(\mathbf{x})$.

More general, we denote by $\mathcal{F}_{n_0,n_1}^{\mathsf{abp}}$ the class of functions $f: \mathbb{Z}_q^{n_0} \to \mathbb{Z}_q^{n_1}$ that evaluate an ABP in each coordinate.

2.4 Dynamic Decentralized Functional Encryption

In this section we recall the notion of *Dynamic Decentralized Functional Encryption* (DDFE). This notion was introduced first in [CDSG⁺20] and later defined in [AGT21b, Section 6.1] as a special case of the Multi-Party Functional Encryption notion. Let $\{ID_{\lambda}\}_{\lambda \in \mathbb{N}}$, $\{\mathcal{K}_{\lambda}\}_{\lambda \in \mathbb{N}}$, $\{\mathcal{M}_{\lambda}\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{R}_{\lambda}\}_{\lambda \in \mathbb{N}}$ be sequences of identity, key, message and output spaces, respectively, and $\mathcal{K}_{\lambda} = \mathcal{K}_{\lambda, \text{pri}} \times \mathcal{K}_{\lambda, \text{pub}}$, $\mathcal{M}_{\lambda} = \mathcal{M}_{\lambda, \text{pri}} \times \mathcal{M}_{\lambda, \text{pub}}$ consist of a private and a public component each. We consider a functionality

$$f^{\mathsf{dyn}} = \left\{ f_{\lambda}^{\mathsf{dyn}} \colon \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_{\lambda} \times \mathcal{K}_{\lambda})^n \times \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_{\lambda} \times \mathcal{M}_{\lambda})^n \to \mathcal{R}_{\lambda} \right\}_{\lambda \in \mathbb{N}}$$

Definition 4 (DDFE Syntax). A DDFE scheme FE for a functionality $f^{dyn} = \{f_{\lambda}^{dyn}\}_{\lambda \in \mathbb{N}}$ consists of five efficient algorithms with the following syntax:

 $\mathsf{GSetup}(1^{\lambda}) \to \mathsf{PP}$: On input the security parameter 1^{λ} this algorithm outputs a public parameter PP . The other algorithms implicitly take PP .

 $\mathsf{LSetup}(i) \to (\mathsf{PK}_i, \mathsf{SK}_i)$: On input an identity $i \in \mathsf{ID}_{\lambda}$, this algorithm outputs a pair of a public key PK_i and a corresponding secret key SK_i . The following three algorithms implicitly take PK_i .

- $\mathsf{KeyGen}(\mathsf{SK}_i, k_i) \to \mathsf{DK}_i$: On input a secret key SK_i and $k_i \in \mathcal{K}_\lambda$, this algorithm outputs a decryption key DK_i .
- $\mathsf{Enc}(\mathsf{SK}_i, m_i) \to \mathsf{CT}_i$: On input a secret key SK_i and $m \in \mathcal{M}_{\lambda}$, this algorithm outputs a ciphertext CT_i .
- $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in\mathcal{U}_K},\{\mathsf{CT}_i\}_{i\in\mathcal{U}_M})\to d\in\mathcal{R}_{\lambda}: \text{ On input a set of decryption keys }\{\mathsf{DK}_i\}_{i\in\mathcal{U}_K} \text{ and a set of } ciphertext \ \{\mathsf{CT}_i\}_{i\in\mathcal{U}_M} \text{ for }\mathcal{U}_K,\mathcal{U}_M\subseteq\mathsf{ID}_{\lambda}, \text{ this algorithm outputs either an element in }\mathcal{R}_{\lambda}.$

Correctness. FE is *correct* if for all $\lambda \in \mathbb{N}$, sets $\mathcal{U}_K, \mathcal{U}_M \subseteq \mathsf{ID}_\lambda$, keys $\{(i, k_i)\}_{i \in \mathcal{U}_K} \in \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_\lambda \times \mathcal{K}_\lambda)^n$, and inputs $\{(i, m_i)\}_{i \in \mathcal{U}_M} \in \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_\lambda \times \mathcal{M}_\lambda)$, we have

$$\Pr \begin{bmatrix} d = f_{\lambda}^{\mathsf{dyn}}(\{(i,k_i)\}_{i \in \mathcal{U}_K}, \\ \{(i,m_i)\}_{i \in \mathcal{U}_M}) \\ & \quad \forall i \in \mathcal{U}_K \cup \mathcal{U}_M \colon (\mathsf{PK}_i,\mathsf{SK}_i) \leftarrow \mathsf{LSetup}(\mathsf{PP}) \\ & \quad \forall i \in \mathcal{U}_K \colon \mathsf{DK}_i \leftarrow \mathsf{KeyGen}(\mathsf{SK}_i,k_i) \\ & \quad \forall i \in \mathcal{U}_M \colon \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{SK}_i,m_i) \\ & \quad d \coloneqq \mathsf{Dec}(\{\mathsf{DK}_i\}_{i \in \mathcal{U}_K}, \{\mathsf{CT}_i\}_{i \in \mathcal{U}_M}) \end{bmatrix} = 1$$

where the probability is taken over the random coins of the algorithms.

Security. We define partially function-hiding security for DDFE as follows.

Definition 5 (DDFE Security). Let $xxx \in \{stat, dyn\}, yyy \in \{sel, sadap, adap\}, zzz \in \{sym, asym\}.$ Given a PPT adversary \mathcal{A} against a DDFE scheme FE for a functionality $f^{dyn} = \{f_{\lambda}^{dyn}\}_{\lambda \in \mathbb{N}}$, we define the experiment $\mathbf{Exp}_{\mathsf{FE}, f^{dyn}, \mathcal{A}}^{\mathsf{ddfe}-b}(1^{\lambda})$ as shown in Figure 1. W.l.o.g., we assume that each *i* is queried at most once to \mathcal{O} HonestGen and \mathcal{O} Corrupt, and that a query \mathcal{O} Corrupt(i) is always preceded by a query \mathcal{O} HonestGen(i). We recall that for the queries to \mathcal{O} KeyGen and \mathcal{O} Enc, namely $(i, k_i^{(0)}, k_i^{(1)})$ and $(i, m_i^{(0)}, m_i^{(1)})$, there are private parts $k_{i,\text{pri}}^{(b)}, m_{i,\text{pri}}^{(b)}$ and public parts $k_{i,\text{pub}}^{(b)}, m_{i,\text{pub}}^{(b)}$ in the keys as well as in the messages. We always require $m_{i,\text{pub}}^{(0)} = m_{i,\text{pub}}^{(1)} =: m_{i,\text{pub}}$ and $k_{i,\text{pub}}^{(0)} = k_{i,\text{pub}}^{(1)} =: k_{i,\text{pub}}$ because the public data is not hidden.

The adversary \mathcal{A} is admissible with respect to $\mathcal{C}, \mathcal{H}, \mathcal{Q}_{enc}, \mathcal{Q}_{key}, denoted by <math>\mathsf{adm}(\mathcal{A}) = 0$, if the following conditions are satisfied. Otherwise, we say that \mathcal{A} is not admissible and write $\operatorname{adm}(\mathcal{A}) = 1$.

- 1. There are no sets $\mathcal{U}_K, \mathcal{U}_M \subseteq \mathsf{ID}_\lambda$ such that there exist sequences $\{(i, k_i^{(0)}, k_i^{(1)})\}_{i \in \mathcal{U}_K}, \{(i, m_i^{(0)}, m_i^{(1)})\}_{i \in \mathcal{U}_M}\}$ that satisfy all the conditions:
- For all i ∈ U_K, (i, k_i⁽⁰⁾, k_i⁽¹⁾) ∈ Q_{key} or [k_i⁽⁰⁾ = k_i⁽¹⁾ and i ∈ C].
 For all i ∈ U_M, (i, m_i⁽⁰⁾, m_i⁽¹⁾) ∈ Q_{enc} or [m_i⁽⁰⁾ = m_i⁽¹⁾ and i ∈ C].
 f_λ^{dyn}({(i, k_i⁽⁰⁾)}_{i∈U_K}, {(i, m_i⁽⁰⁾)}_{i∈U_M}) ≠ f_λ^{dyn}({(i, k_i⁽¹⁾)}_{i∈U_K}, {(i, m_i⁽¹⁾)}_{i∈U_M}).
 2. If xxx = stat, then the adversary first generates a set S of honest users. After that it submits all queries to OCorrupt in one shot.
- 3. If yyy = sel, then the adversary first generates a set S of honest users. After that it submits all queries to $\mathcal{O}Enc$ and $\mathcal{O}KeyGen$ in one shot. If yyy = sadap, then the adversary cannot call $\mathcal{O}Enc$ anymore after submitting the first query to OKeyGen.
- 4. If zzz = sym, then for $i \in \mathcal{C}$ all queries $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{key}$ and $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{enc}$ satisfy $k_i^{(0)} = k_i^{(1)}$ and $m_i^{(0)} = m_i^{(1)}$, respectively.⁴

We say that FE is xxx-yyy-zzz-secure if for all PPT adversaries \mathcal{A} ,

$$\mathbf{Exp}^{\mathsf{ddfe-0}}_{\mathsf{FE},f^{\mathsf{dyn}},\mathcal{A}}(1^{\lambda}) \approx_{c} \mathbf{Exp}^{\mathsf{ddfe-1}}_{\mathsf{FE},f^{\mathsf{dyn}},\mathcal{A}}(1^{\lambda})$$

$ \begin{array}{l} \begin{array}{l} \mbox{Initialize}(1^{\lambda}):\\ \hline {\mathcal{C},\mathcal{H},\mathcal{Q}_{enc},\mathcal{Q}_{key}} \leftarrow \varnothing \\ \mbox{PP} \leftarrow \mbox{GSetup}(1^{\lambda}) \\ \mbox{Return PP} \end{array} \end{array} $	$\frac{\mathcal{O}KeyGen(i, k_i^{(0)}, k_i^{(1)}):}{\mathcal{Q}_{key} \leftarrow \mathcal{Q}_{key} \cup \{(i, k_i^{(0)}, k_i^{(1)})\}}$ Return $DK_i \leftarrow KeyGen(SK_i, k_i^{(b)})$
$ \begin{array}{l} \mathcal{O}HonestGen(i):\\ \hline (PK_i,SK_i) \leftarrow LSetup(PP)\\ \mathcal{H} \leftarrow \mathcal{H} \cup \{i\}; \ \mathrm{return} \ PK_i \end{array} $	$\frac{\mathcal{O}Enc(i, m_i^{(0)}, m_i^{(1)}):}{\overline{\mathcal{Q}_{enc}} \leftarrow \mathcal{Q}_{enc} \cup \{(i, m_i^{(0)}, m_i^{(1)})\}}$ Return $CT_i \leftarrow Enc(SK_i, m_i^{(b)})$
$ \frac{\mathcal{O}Corrupt(i):}{\mathcal{H} \leftarrow \mathcal{H} \setminus \{i\};} \mathcal{C} \leftarrow \mathcal{C} \cup \{i\} $ Return SK _i	$\frac{Finalize(b'):}{\text{If }adm(\mathcal{A}) = 0, \text{ return } \beta \leftarrow (b' \stackrel{?}{=} b)}{\text{Else, return a random bit } \beta \stackrel{\$}{\leftarrow} \{0, 1\}}$

Fig. 1: Security game $\mathbf{Exp}_{\mathsf{FE}, f^{\mathsf{dyn}}, \mathcal{A}}^{\mathsf{ddfe}-b}(1^{\lambda})$ for Definition 5

⁴ The symmetric setting allows proving security in the case where SK_i cannot only be used to encrypt/generate keys but also to decrypt/decode CT_i/DK_i .

Functionalities. First, we define the functionalities f^{dyn-ip} and $f^{dyn-fh-ip}$ of bounded-norm innerproducts with standard or function-hiding security.

Definition 6 (Inner-Product Functionality). For $\lambda \in \mathbb{N}$, let $\mathsf{Tag}_{\lambda} = \mathsf{ID}_{\lambda} = \{0, 1\}^{\mathsf{poly}(\lambda)}$, $\mathcal{R}_{\lambda} = \mathbb{Z}, \ \mathcal{K}_{\lambda, \text{pub}} = [-B; B]^N \times 2^{\text{ID}_{\lambda}} \times \text{Tag}_{\lambda}, \ \mathcal{M}_{\lambda, \text{pub}} = 2^{\text{ID}_{\lambda}} \times \text{Tag}_{\lambda}, \ \mathcal{K}_{\lambda, \text{pri}} = \{\top\} \ and \ \mathcal{M}_{\lambda, \text{pri}} = [-B; B]^N \ for \ polynomials \ B = B(\lambda) \ and \ N = N(\lambda) \colon \mathbb{N} \to \mathbb{N}.$ The inner-product functionality $f^{\mathsf{dyn-ip}} = \{f_{\lambda}^{\mathsf{dyn-ip}}\}_{\lambda \in \mathbb{N}} \text{ is defined via}$

$$f_{\lambda}^{\mathsf{dyn-ip}}\big(\{(i,k_i)\}_{i\in\mathcal{U}_K},\{(i,m_i)\}_{i\in\mathcal{U}_M}\big) = \begin{cases} \sum_{i\in\mathcal{U}} \langle \mathbf{x}_i,\mathbf{y}_i \rangle & \text{if condition (*) holds} \\ \bot & \text{otherwise} \end{cases}$$

for all $\lambda \in \mathbb{N}$, where condition (*) holds if $\mathcal{U}_K = \mathcal{U}_M$ (in which case we define $\mathcal{U} \coloneqq \mathcal{U}_K$) and there exist tag, tag-f \in Tag, such that for each $i \in \mathcal{U}$

- k_i is of the form (k_{i,pri} = ⊤, k_{i,pub} = (y_i, U, tag-f)), and
 m_i is of the form (m_{i,pri} = x_i, m_{i,pub} = (U, tag)).

The functionality $f^{\mathsf{dyn-fh-ip}} = \{f_{\lambda}^{\mathsf{dyn-fh-ip}}\}_{\lambda \in \mathbb{N}}$ is the same as $f^{\mathsf{dyn-ip}}$ except that we set $\mathcal{K}_{\lambda,\mathrm{pub}} = 2^{\mathsf{ID}_{\lambda}} \times \mathsf{Tag}_{\lambda}, \ \mathcal{K}_{\lambda,\mathrm{pri}} = [-B;B]^N$ and condition (*) holds if $\mathcal{U}_K = \mathcal{U}_M$ (in which case we define $\mathcal{U} \coloneqq \mathcal{U}_K$) and there exist tag, tag-f $\in \mathsf{Tag}_{\lambda}$ such that for each $i \in \mathcal{U}$

- k_i is of the form $(k_{i,\text{pri}} = \mathbf{y}_i, k_{i,\text{pub}} = (\mathcal{U}, \text{tag-f}))$, and
- m_i is of the form $(m_{i,\text{pri}} = \mathbf{x}_i, m_{i,\text{pub}} = (\mathcal{U}, \text{tag})).$

Second, we define the functionality $f^{dyn-ab-aws}$ of Attribute-Based Attribute-Weighted Sums (AB-AWS) which is a generalization of the previous inner-product functionality.

Definition 7 (Attribute-Based Attribute-Weighted Sum Functionality). Let $\mathbb{G} = \{\mathbb{G}_{\lambda} =$ $(\mathbb{G}_{1,\lambda},\mathbb{G}_{2,\lambda},\mathbb{G}_{t,\lambda},g_{1,\lambda},g_{2,\lambda},g_{t,\lambda},e_{\lambda},q_{\lambda})\}_{\lambda\in\mathbb{N}}$ be a sequence of pairing groups. For $\lambda\in\mathbb{N}$, let $\mathsf{Tag}_{\lambda}=$ $\mathsf{ID}_{\lambda} = \{0,1\}^{\mathrm{poly}(\lambda)}, \ \mathcal{R}_{\lambda} = \mathbb{G}_{\mathsf{t},\lambda}, \ \mathcal{K}_{\lambda,\mathrm{pub}} = \mathcal{F}_{n_{0}',1}^{\mathsf{abp}} \times \mathcal{F}_{n_{0},n_{1}}^{\mathsf{abp}} \times 2^{\mathsf{ID}_{\lambda}} \times \mathsf{Tag}_{\lambda}, \ \mathcal{M}_{\lambda,\mathrm{pub}} = (\mathbb{Z}_{q_{\lambda}}^{n_{0}'} \cup \{\star\}) \times \mathbb{E}_{q_{\lambda}'} \times \mathbb{E$ $\bigcup_{N \in \mathbb{N}} (\mathbb{Z}_q^{n_0})^N \times 2^{\mathsf{ID}_{\lambda}} \times \mathsf{Tag}_{\lambda}, \, \mathcal{K}_{\lambda, \mathsf{pri}} = \{\top\} \text{ and } \mathcal{M}_{\lambda, \mathsf{pri}} = \bigcup_{N' \in \mathbb{N}} (\mathbb{Z}_q^{n_1})^{N'} \text{ for polynomials } n'_0 = n'_0(\lambda), \, n_0 = n_0(\lambda), \, n_1 = n_1(\lambda), \, N = N(\lambda) \text{ and } N' = N'(\lambda) \colon \mathbb{N} \to \mathbb{N}. \text{ The AB-AWS functionality}$ $f^{\mathsf{dyn-ab-aws}} = \{f^{\mathsf{dyn-ab-aws}}_{\lambda}\}_{\lambda \in \mathbb{N}} \text{ is defined via}$

$$\begin{aligned} f_{\lambda}^{\mathsf{dyn-ab-aws}} \big(\{(i,k_i)\}_{i \in \mathcal{U}_K}, \{(i,m_i)\}_{i \in \mathcal{U}_M} \big) = \\ & \left\{ \begin{aligned} \left[\sum_{i \in \mathcal{U}} \sum_{j \in [N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle \right]_{\mathsf{t}} & \text{if condition (*) holds} \\ \bot & \text{otherwise} \end{aligned} \right. \end{aligned}$$

for all $\lambda \in \mathbb{N}$, where condition (*) holds if $\mathcal{U}_K = \mathcal{U}_M$ (in which case we define $\mathcal{U} \coloneqq \mathcal{U}_K$) and there exist tag, tag-f $\in \mathsf{Tag}_{\lambda}$ such that for each $i \in \mathcal{U}$

- k_i is of the form $(k_{i,\text{pri}} = \top, k_{i,\text{pub}} = (g_i, h_i, \mathcal{U}, \mathsf{tag-f})),$
- m_i is of the form $(m_{i,\text{pri}} = \{\mathbf{z}_{i,j}\}_{j \in [N'_i]}, m_{i,\text{pub}} = (\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_{j \in [N_i]}, \mathcal{U}, \text{tag}))$ such that $N'_i = N_i$, and
- for all $i \in [n]$, $g_i(\mathbf{y}_i) = 0$ or $\mathbf{y}_i = \star$.

Definition 8 (Legitimate Queries for AB-AWS). For a set $\mathcal{U} \subseteq \mathsf{ID}$, we denote by $T(\mathcal{U})$ the set of all function tags tag-f such that there exists a query $(j, k_j^{(0)}, k_j^{(1)}) \in \mathcal{Q}_{key}$ with $j \in \mathcal{H}$ and $k_{j, pub} =$ $(g_j, h_j, \mathcal{U}, \mathsf{tag-f})$. An encryption query $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ with $m_{i, \mathrm{pub}} = (\mathbf{y}_i, \{\mathbf{x}_{i,j'}\}_{j' \in [N_i]}, \mathcal{U}, \mathsf{tag})$ is legitimate if $m_i^{(0)} = m_i^{(1)}$ or, for all $j \in \mathcal{U} \cap \mathcal{H}$ and tag-f $\in T(\mathcal{U})$, there exist $(j, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{kev}$ with $k_{j,\text{pub}} = (g_j, h_j, \mathcal{U}, \mathsf{tag-f})$ and $(j, m_j^{(0)}, m_j^{(1)}) \in \mathcal{Q}_{\mathsf{enc}}$ with $m_{j,\text{pub}} = (\mathbf{y}_j, \{\mathbf{x}_{j,j'}\}_{j' \in [N_i]}, \mathcal{U}, \mathsf{tag})$ such that $g_i(\mathbf{y}_i) = 0$. Furthermore, a DDFE for AB-AWS is secure against legitimate queries if the scheme is secure against all admissible (i.e. $\operatorname{adm}(\mathcal{A}) = 0$ as per Definition 5) adversaries \mathcal{A} that submit only legitimate encryption queries.

A DDFE is called a single-client FE scheme if $ID_{\lambda} = \{\top\}$ is a singleton for all $\lambda \in \mathbb{N}$. The functionality of Attribute-Based Attribute-Weighted Sums with Inner Products (AB-AWSw/IP) for single-client FE schemes is defined as follows.

Definition 9 (AB-AWSw/IP). Let $\mathbb{G} = \{\mathbb{G}_{\lambda} = (\mathbb{G}_{1,\lambda}, \mathbb{G}_{2,\lambda}, \mathbb{G}_{t,\lambda}, g_{1,\lambda}, g_{2,\lambda}, g_{t,\lambda}, e_{\lambda}, q_{\lambda})\}$ be a sequence of pairing groups. For $\lambda \in \mathbb{N}$, let $\mathcal{R}_{\lambda} = \mathbb{G}_{t,\lambda}$, $\mathcal{K}_{\lambda,\text{pub}} = \mathcal{F}_{n'_0,1}^{\mathsf{abp}} \times \mathcal{F}_{n_0,n_1}^{\mathsf{abp}}$, $\mathcal{K}_{\lambda,\text{pri}} = \mathbb{G}_{2,\lambda}^m$, $\mathcal{M}_{\lambda,\text{pub}} = (\mathbb{Z}_{q_{\lambda}}^{n'_0} \cup \{\star\}) \times \bigcup_{N \in \mathbb{N}} (\mathbb{Z}_q^{n_0})^N$ and $\mathcal{M}_{\lambda,\text{pri}} = \bigcup_{N' \in \mathbb{N}} (\mathbb{Z}_q^{n_1})^{N'} \times \mathbb{G}_{1,\lambda}^m$. The functionality $f^{\mathsf{ab-aws-ip}} = \{f_{\lambda}^{\mathsf{ab-aws-ip}}\}_{\lambda \in \mathbb{N}}$ is defined via

$$f_{\lambda}^{\text{ab-aws-ip}}(k,m) = \begin{cases} \left[\sum_{j \in [N]} \langle h(\mathbf{x}_j), \mathbf{z}_j \rangle + \langle \mathbf{p}, \mathbf{q} \rangle \right]_{t} & \text{if (*) holds} \\ \bot & \text{otherwise} \end{cases}$$

for all $\lambda \in \mathbb{N}$ and condition (*) is satisfied if

- *m* is of the form $(m_{\text{pri}} = (\{\mathbf{z}_j\}_{j \in [N']}, [[\mathbf{p}]]_1), m_{\text{pub}} = (\mathbf{y}, \{\mathbf{x}_j\}_{j \in [N]}))$ such that N' = N,
- k is of the form $(k_{pri} = \llbracket \mathbf{q} \rrbracket_2, k_{pub} = (g, h))$, and
- $g(\mathbf{y}) = 0$ or $\mathbf{y} = \star$.

FE for AB-AWSw/IP with sadap-security is known to exist under the $MDDH_k$ assumption and pairings [ATY23].

2.5 Decentralized Multi-Client Functional Encryption

The notion of *Decentralized Multi-Client Functional Encryption* (DMCFE) introduced in [CDG⁺18a] can be identified as a special case of DDFE, where the number n of users is fixed in advanced by a (possibly interactive) global setup and there is no local setting up so that a new user can enter the system. Moreover, for efficiency, prior papers (such as [CDG⁺18a, CDG⁺18b, ABKW19, ABG19, LT19, CDSG⁺20]) considered an additional *key combination* algorithm that, given ndecryption key components {DK_{tag-f,i}}_{i\in[n]} generated for the same tag tag-f, outputs a succinct functional key DK_{tag-f} which can be passed to decryption Dec(DK_{tag-f}, {CT_{tag,i}}_{i\in[n]}). Without loss of generality, the DMCFE notion in this paper implicitly includes the key combination algorithm in the decryption algorithm and whenever we refer to other existing DMCFE schemes, they are syntactically understood as such. The formal definition of DMCFE that is used in this paper is given below.

Let $\{\mathsf{Tag}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathcal{K}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathcal{M}_{\lambda}\}_{\lambda\in\mathbb{N}} \text{ and } \{\mathcal{R}_{\lambda}\}_{\lambda\in\mathbb{N}} \text{ be sequences of tag, key, message and output spaces, respectively, and <math>\mathcal{K}_{\lambda} = \mathcal{K}_{\lambda,\mathrm{pri}} \times \mathcal{K}_{\lambda,\mathrm{pub}}, \mathcal{M}_{\lambda} = \mathcal{M}_{\lambda,\mathrm{pri}} \times \mathcal{M}_{\lambda,\mathrm{pub}} \text{ consist of a private and a public component each. We consider a functionality } f = \{f_{\lambda,n}: \mathcal{K}_{\lambda}^n \times \mathcal{M}_{\lambda}^n \to \mathcal{R}_{\lambda}\}_{\lambda,n\in\mathbb{N}}.$

Definition 10 (DMCFE Syntax). A DMCFE scheme FE for the functionality $f = \{f_{\lambda,n}\}_{\lambda,n\in\mathbb{N}}$ consists of the four efficient algorithms defined below:

- $\mathsf{Setup}(1^{\lambda}, 1^n) \to (\mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]})$: This is a protocol between the n clients that eventually generate their own secret keys SK_i , as well as the public parameter PP . The other algorithms implicitly take PP .
- $\mathsf{KeyGen}(\mathsf{SK}_i, \mathsf{tag-f}, y_i) \to \mathsf{DK}_{\mathsf{tag-f},i}$: On input a secret key SK_i , a tag $\mathsf{tag-f} \in \mathsf{Tag}_{\lambda}$, and $k_i \in \mathcal{K}_{\lambda}$, this algorithm outputs a decryption key $\mathsf{DK}_{\mathsf{tag-f},i}$.
- $\mathsf{Enc}(\mathsf{SK}_i, \mathsf{tag}, x_i) \to \mathsf{CT}_{\mathsf{tag},i}$: On input a secret key SK_i , a tag $\mathsf{tag} \in \mathsf{Tag}_\lambda$ and $x_i \in \mathcal{M}_\lambda$, this algorithm outputs a ciphertext $\mathsf{CT}_{\mathsf{tag},i}$.
- $Dec(\{DK_{tag-f,i}\}_{i\in[n]}, \{CT_{tag,i}\}_{i\in[n]}) \rightarrow d: \text{ On input a set of functional decryption keys } \{DK_{tag-f,i}\}_{i\in[n]} all \text{ generated for the same tag tag-f and a set of ciphertexts } \{CT_{tag,i}\}_{i\in[n]} all \text{ generated for the same tag tag, this algorithm outputs an element } d \in \mathcal{R}_{\lambda}.$

Correctness. FE is *correct* if for all $\lambda, n \in \mathbb{N}$, all tags tag-f, tag $\in \mathsf{Tag}_{\lambda}$ and all inputs $\{k_i\}_{i \in [n]} \subseteq \mathcal{K}_{\lambda}$ and $\{m_i\}_{i\in[n]} \subseteq \mathcal{M}_{\lambda}$, we have

$$\Pr \begin{bmatrix} d = f_{\lambda,n}(\{k_i\}_{i \in [n]}, \\ \{m_i\}_{i \in [n]}) & \forall i \in [n] : \mathsf{DK}_{\mathsf{tag-f},i} \leftarrow \mathsf{KeyGen}(\mathsf{SK}_i, \mathsf{tag-f}, k_i) \\ \forall i \in [n] : \mathsf{CT}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{SK}_i, \mathsf{tag}, m_i) \\ d \coloneqq \mathsf{Dec}(\{\mathsf{DK}_{\mathsf{tag-f},i}\}_{i \in [n]}, \{\mathsf{CT}_{\mathsf{tag},i}\}_{i \in [n]}) \end{bmatrix} = 1$$

where the probability is taken over the random coins of the algorithms.

Security. We define security for DMCFE as follows.

Definition 11 (DMCFE Security). Let $xxx \in \{stat, dyn\}, yyy \in \{sel, sadap, adap\}, zzz \in \{sel, sadap, adap\}$ {sym, asym}. Given a PPT adversary A against a DMCFE scheme FE for a functionality f = $\{f_{\lambda,n}\}_{\lambda,n\in\mathbb{N}}$, we define the experiment $\mathbf{Exp}_{\mathsf{FE},f,\mathcal{A}}^{\mathsf{dmcfe}-b}(1^{\lambda})$ as shown in Figure 2. We recall that for the queries to \mathcal{O} KeyGen and \mathcal{O} Enc, namely $(i, \mathsf{tag-f}, k_i^{(0)}, k_i^{(1)})$ and $(i, \mathsf{tag}, m_i^{(0)}, m_i^{(1)})$, there are private parts $k_{i,\text{pri}}^{(b)}, m_{i,\text{pri}}^{(b)}$ and public parts $k_{i,\text{pub}}^{(b)}, m_{i,\text{pub}}^{(b)}$ in the keys as well as in the messages. We always require $m_{i,\text{pub}}^{(0)} = m_{i,\text{pub}}^{(1)} =: m_{i,\text{pub}}$ and $k_{i,\text{pub}}^{(0)} = k_{i,\text{pub}}^{(1)} =: k_{i,\text{pub}}$ because the public data is not hidden. The adversary \mathcal{A} is admissible with respect to $\mathcal{C}, \mathcal{Q}_{enc}, \mathcal{Q}_{key}$, denoted by $\operatorname{adm}(\mathcal{A}) = 0$, if the

following conditions are satisfied. Otherwise, we say that \mathcal{A} is not admissible and write $\mathsf{adm}(\mathcal{A}) = 1$.

- 1. There are no tags tag-f, tag $\in \mathsf{Tag}_{\lambda}$ such that there exist sequences $\{(i, \mathsf{tag-f}, k_i^{(0)}, k_i^{(1)})\}_{i \in [n]}, k_i^{(0)}\}$ $\{(i, tag, m_i^{(0)}, m_i^{(1)})\}_{i \in [n]}$ that satisfy all the conditions:

 - $\begin{array}{l} \bullet \ For \ all \ i \in [n], \ (i, \mathsf{tag-f}, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{key}} \ or \ [k_i^{(0)} = k_i^{(1)} \ and \ i \in \mathcal{C}]. \\ \bullet \ For \ all \ i \in [n], \ (i, \mathsf{tag}, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{enc}} \ or \ [m_i^{(0)} = m_i^{(1)} \ and \ i \in \mathcal{C}]. \\ \bullet \ f_{\lambda}(\{(i, k_i^{(0)})\}_{i \in [n]}, \{(i, m_i^{(0)})\}_{i \in [n]}) \neq f_{\lambda}(\{(i, k_i^{(1)})\}_{i \in [n]}, \{(i, m_i^{(1)})\}_{i \in [n]}). \end{array}$
- 2. If xxx = stat, then the adversary submits all queries to $\mathcal{O}Corrupt$ up front in one shot.
- 3. If yyy = sel, then the adversary submits all queries to $\mathcal{O}Enc$ and $\mathcal{O}KeyGen$ up front in one shot. If yyy = sadap, then the adversary cannot call OEnc anymore after submitting the first query to OKeyGen.
- 4. If zzz = sym, then for $i \in \mathcal{C}$ all queries $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{key}$ and $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{enc}$ satisfy $k_i^{(0)} = k_i^{(1)}$ and $m_i^{(0)} = m_i^{(1)}$, respectively.⁵

We say that FE is xxx-yyy-zzz-secure if for all PPT adversaries \mathcal{A} ,

$$\mathbf{Exp}_{\mathsf{FE},f,\mathcal{A}}^{\mathsf{dmcfe-0}}(1^{\lambda}) \approx_{c} \mathbf{Exp}_{\mathsf{FE},f,\mathcal{A}}^{\mathsf{dmcfe-1}}(1^{\lambda})$$
 .

Functionalities. We give the static versions of the inner-product and attribute-based attributeweighted sums functionalities introduced in Definitions 6 and 7.

Definition 12 (Inner Product Functionality). For $\lambda \in \mathbb{N}$, let $\mathcal{R}_{\lambda} = \mathbb{Z}$, $\mathcal{K}_{\lambda, \text{pub}} = \mathcal{M}_{\lambda, \text{pri}} =$ $[-B;B]^N$ and $\mathcal{K}_{\lambda,\text{pri}} = \mathcal{M}_{\lambda,\text{pub}} = \{\top\}$ for polynomials $B = B(\lambda)$ and $N = N(\lambda) \colon \mathbb{N} \to \mathbb{N}$. The functionality $f^{ip} = \{f_{\lambda,n}^{ip}\}_{\lambda,n \in \mathbb{N}}$ for standard security is defined via

$$f_{\lambda,n}^{\mathsf{ip}}\big(\{k_i = (\top, \mathbf{y}_i)\}_{i \in [n]}, \{m_i = (\mathbf{x}_i, \top)\}_{i \in [n]}\big) = \sum_{i \in [n]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle$$

⁵ A recent work [NPP23] studies a stronger security notion that removes this condition for (D)MCFE.

$ \begin{array}{l} \underset{\mathcal{C}, \mathcal{Q}_{enc}, \mathcal{Q}_{key} \leftarrow \varnothing}{Initialize(1^{\lambda}, 1^{n}):} \\ (PP, \{SK_i\}_{i \in [n]}) \leftarrow Setup(1^{\lambda}) \\ \operatorname{Return} PP \end{array} $	$\frac{\mathcal{O}Enc(i, tag, m_i^{(0)}, m_i^{(1)}):}{\mathcal{Q}_{enc} \leftarrow \mathcal{Q}_{enc} \cup \{(i, tag, m_i^{(0)}, m_i^{(1)})\}}$ Return $CT_i \leftarrow Enc(SK_i, tag, m_i^{(b)})$
$\frac{\mathcal{O}KeyGen(i,tag-f,k_i^{(0)},k_i^{(1)}):}{\mathcal{Q}_{kev} \leftarrow \mathcal{Q}_{kev} \cup \{(i,tag-f,k_i^{(0)},k_i^{(1)})\}}$	$\frac{\mathcal{O}Corrupt(i):}{\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}; \text{ return } SK_i}$
Return $DK_i \leftarrow KeyGen(SK_i, tag-f, k_i^{(b)})$	$\frac{\text{Finalize}(b')}{\text{If } \text{adm}(A)} = 0 \text{ actum} \theta \left(\frac{b'}{2} \right)$
	Else, return a random bit $\beta \stackrel{\text{\ensuremath{\&}}}{=} \{0, 1\}$

Fig. 2: Security game $\mathbf{Exp}_{\mathsf{FE},f,\mathcal{A}}^{\mathsf{dmcfe},b}(1^{\lambda})$ for Definition 11

for all $\lambda, n \in \mathbb{N}$. The functionality $f^{\text{fh-ip}} = \{f_{\lambda,n}^{\text{fh-ip}}\}_{\lambda,n\in\mathbb{N}}$ for function-hiding security is defined as f^{ip} except that we set $\mathcal{K}_{\lambda,\text{pub}} = \{\top\}, \mathcal{K}_{\lambda,\text{pri}} = [-B; B]^N$ and

$$f_{\lambda,n}^{\mathsf{fh-ip}}\big(\{k_i = (\mathbf{y}_i, \top)\}_{i \in [n]}, \{m_i = (\mathbf{x}_i, \top)\}_{i \in [n]}\big) = \sum_{i \in [n]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle$$

for all $\lambda, n \in \mathbb{N}$.

Definition 13 (Attribute-Based Attribute-Weighted Sum Functionality). Let $\mathbb{G} = \{\mathbb{G}_{\lambda} = (\mathbb{G}_{1,\lambda}, \mathbb{G}_{2,\lambda}, \mathbb{G}_{t,\lambda}, g_{1,\lambda}, g_{2,\lambda}, g_{t,\lambda}, e_{\lambda}, q_{\lambda})\}_{\lambda \in \mathbb{N}}$ be a sequence of pairing groups. For $\lambda \in \mathbb{N}$, let $\mathcal{R}_{\lambda} = \mathbb{G}_{t,\lambda}$, $\mathcal{K}_{\lambda,\text{pub}} = \mathcal{F}_{n'_{0},1}^{\text{abp}} \times \mathcal{F}_{n_{0},n_{1}}^{\text{abp}}$, $\mathcal{K}_{\lambda,\text{pri}} = \{\top\}$, $\mathcal{M}_{\lambda,\text{pub}} = (\mathbb{Z}_{q_{\lambda}}^{n'_{0}} \cup \{\star\}) \times \bigcup_{N \in \mathbb{N}} (\mathbb{Z}_{q}^{n_{0}})^{N}$ and $\mathcal{M}_{\lambda,\text{pri}} = \bigcup_{N' \in \mathbb{N}} (\mathbb{Z}_{q}^{n_{1}})^{N'}$. The functionality $f^{\text{ab-aws}} = \{f_{\lambda,n}^{\text{ab-aws}}\}_{\lambda,n \in \mathbb{N}}$ is defined via

$$f_{\lambda,n}^{\mathsf{ab-aws}}\big(\{k_i\}_{i\in[n]},\{m_i\}_{i\in[n]}\big) = \begin{cases} \left[\sum_{i\in[n]}\sum_{j\in[N_i]}\langle h_i(\mathbf{x}_{i,j}),\mathbf{z}_{i,j}\rangle\right]_{\mathsf{t}} & if (*) \ holds \\ \bot & otherwise \end{cases}$$

for all $\lambda, n \in \mathbb{N}$ and condition (*) is satisfied if

- m_i is of the form $(m_{i,\text{pri}} = \{\mathbf{z}_{i,j}\}_{j \in [N'_i]}, m_{i,\text{pub}} = (\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_{j \in [N_i]}))$ such that $N'_i = N_i$,
- k_i is of the form $(k_{i,\text{pri}} = \top, k_{i,\text{pub}} = (g_i, h_i))$, and
- for all $i \in [n]$, $g_i(\mathbf{y}_i) = 0$ or $\mathbf{y}_i = \star$.

Definition 14 (Legitimate Queries for AB-AWS). We denote by T the set of all function tags tag-f such that there exists a query $(j, tag-f, k_j^{(0)}, k_j^{(1)}) \in \mathcal{Q}_{key}$ with $j \in \mathcal{H}$. An encryption query $\mathcal{O}\mathsf{Enc}(i, tag, m_i^{(0)}, m_i^{(1)})$ with $m_{i, pub} = (\mathbf{y}_i, \{\mathbf{x}_{i,j'}\}_{j' \in [N_i]})$ is legitimate if $m_i^{(0)} = m_i^{(1)}$ or, for all $j \in [n] \setminus \mathcal{C}$ and tag-f $\in T$, there exist $(j, tag-f, k_j^{(0)}, k_j^{(1)}) \in \mathcal{Q}_{key}$ with $k_{j, pub} = (g_j, h_j)$ and $(j, tag, m_j^{(0)}, m_j^{(1)}) \in \mathcal{Q}_{enc}$ with $m_{j, pub} = (\mathbf{y}_j, \{\mathbf{x}_{j,j'}\}_{j' \in [N_j]})$ such that $g_j(\mathbf{y}_j) = 0$. Furthermore, a DMCFE for AB-AWS is secure against legitimate queries if the scheme is secure against all admissible (i.e. $\operatorname{adm}(\mathcal{A}) = 0$ as per Definition 11) adversaries \mathcal{A} that submit only legitimate encryption queries.

2.6 Attribute-Based and Identity-Based Encryption

We recall the definition of Attribute-Based Encryption [SW05].

Definition 15 (Attribute-Based Encryption (ABE)). Let $\mathcal{M} = {\mathcal{M}_{\lambda}}_{\lambda \in \mathbb{N}}$, $\mathcal{X} = {\mathcal{X}_{\lambda}}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = {\mathcal{Y}_{\lambda}}_{\lambda \in \mathbb{N}}$ be sequences of message, ciphertext-attribute and key-attribute spaces, and let $f = {f_{\lambda}}_{\lambda \in \mathbb{N}}$ be a sequence of predicates where $f_{\lambda} : \mathcal{X}_{\lambda} \times \mathcal{Y}_{\lambda} \to {0,1}$ for all $\lambda \in \mathbb{N}$. An ABE scheme ABE for \mathcal{M} and f consists of the four efficient algorithms defined below:

- $\mathsf{Setup}(1^{\lambda}) \to (\mathsf{MPK}, \mathsf{MSK})$: On input the security parameter 1^{λ} , this algorithm outputs a pair of a master public key MPK and a master secret key MSK.
- $\mathsf{KeyGen}(\mathsf{MSK}, y) \to \mathsf{DK}_y$: On input the master secret key MSK and an attribute $y \in \mathcal{Y}_{\lambda}$, this algorithm outputs a decryption key DK_y .
- $\mathsf{Enc}(\mathsf{MPK}, x) \to \mathsf{DK}_x$: On input the master public key MPK an attribute $x \in \mathcal{X}_\lambda$ and a message $\mu \in \mathcal{M}_\lambda$, this algorithm outputs a ciphertext CT_x .
- $\mathsf{Dec}(\mathsf{DK}_y,\mathsf{CT}_x) \to \mu' \lor \bot$: On input a decryption keys DK_y and a ciphertext CT_x , this algorithm outputs an element $\mu' \in \mathcal{M}_\lambda$ or \bot .

Correctness. The ABE scheme ABE is *correct* if for all $\lambda \in \mathbb{N}$, $\mu \in \mathcal{M}_{\lambda}$, $x \in \mathcal{X}_{\lambda}$ and $y \in \mathcal{Y}_{\lambda}$ such that $f_{\lambda}(x, y) = 0$, we have

$$\Pr \begin{bmatrix} \mu = \mu' & \left(\begin{array}{c} (\mathsf{MPK},\mathsf{MSK}) \leftarrow \mathsf{Setup}(1^{\lambda}) \\ \mathsf{DK}_y \leftarrow \mathsf{KeyGen}(\mathsf{MSK},y) \\ \mathsf{CT}_x \leftarrow \mathsf{Enc}(\mathsf{MPK},x,\mu) \\ \mu' \coloneqq \mathsf{Dec}(\mathsf{DK}_y,\mathsf{CT}_x) \end{array} \right] = 1 \ ,$$

where the probability is taken over the random coins of the algorithms. **Security.** We define security for ABE as follows.

Definition 16 (ABE Security). Given a PPT adversary \mathcal{A} against an ABE scheme ABE for a predicate $f = \{f_{\lambda}\}_{\lambda \in \mathbb{N}}$, we define the experiment $\operatorname{Exp}_{\mathsf{ABE},\mathcal{A}}^{\mathsf{abe}-b}(1^{\lambda})$ as shown in Figure 3. The oracle \mathcal{O} KeyGen can be called any (polynomial) number of times whereas the oracle \mathcal{O} Enc can be called only once. We say that FE is secure if for all PPT adversaries \mathcal{A} ,

$$\mathbf{Exp}_{\mathsf{ABE},\mathcal{A}}^{\mathsf{abe}-0}(1^{\lambda}) \approx_{c} \mathbf{Exp}_{\mathsf{ABE},\mathcal{A}}^{\mathsf{abe}-1}(1^{\lambda})$$

$ \begin{array}{l} \underset{\mathcal{Q}\leftarrow\varnothing;\ x_{enc}=\bot}{\text{Initialize}(1^{\lambda},1^{n}):} \\ \hline \mathcal{Q}\leftarrow\varnothing;\ x_{enc}=\bot \\ (\text{MPK},\text{MSK})\leftarrow\text{Setup}(1^{\lambda}) \end{array} $	$\frac{\mathcal{O}Enc(x,\mu^{(0)},\mu^{(1)}):}{x_{enc} \leftarrow x}$ Return $CT_x \leftarrow Enc(MSK,x,\mu^{(b)})$
Return MPK	
	$\frac{\text{Finalize}(b'):}{\frac{1}{16} f(x-x)} = 1 \text{ for all } x \in \mathcal{O}$
$\frac{\mathcal{O}KeyGen(y)}{\mathcal{O}KeyGen(y)}$	If $f(x_{enc}, y) = 1$ for all $y \in \mathcal{Q}$,
$\mathcal{Q}_{\text{key}} \leftarrow \mathcal{Q}_{\text{key}} \cup \{y\}$	return $\beta \leftarrow (b' \doteq b)$
Return $DK_y \leftarrow KeyGen(MSK, y)$	Else, return a random bit $\beta \stackrel{s}{\leftarrow} \{0,1\}$

Fig. 3: Security game $\mathbf{Exp}_{\mathsf{ABE},\mathcal{A}}^{\mathsf{abe}-b}(1^{\lambda})$ for Definition 16

ABE schemes for ABPs are known to exist under the $MDDH_k$ assumption and pairings [LL20]. We define identity-based encryption as a special case of ABE.

Definition 17 (Identity-Based Encryption (IBE)). Let $\mathsf{ID} = \{\mathsf{ID}_{\lambda}\}_{\lambda \in \mathbb{N}}$ be a sequence of identity spaces. An IBE scheme for ID is an ABE for the attribute spaces $\mathcal{X} = \mathcal{Y} = \mathsf{ID}$ and the equality predicates, i.e. $f = \{f_{\lambda}\}_{\lambda \in \mathbb{N}}$ where $f_{\lambda}(x, y) = (x \stackrel{?}{=} y)$ for all $\lambda \in \mathbb{N}$.

2.7 Lockable Obfuscation

We recall the definition of a lockable obfuscator [GKW17, WZ17]. Given polynomials $n = n(\lambda), m = m(\lambda)$ and $d = d(\lambda)$, we denote by $\mathcal{C}_{n,m,d}(\lambda)$ the class of depth $d(\lambda)$ circuits with $n(\lambda)$ bits input and $m(\lambda)$ bits output.

Definition 18 (Lockable Obfuscation). Let $\mathcal{M} = {\mathcal{M}_{\lambda}}_{\lambda \in \mathbb{N}}$ be a sequence of message spaces and ${\mathcal{C}_{n,m,d}(\lambda)}_{\lambda \in \mathbb{N}}$ a sequence of circuit classes. A lockable obfuscator for \mathcal{M} and \mathcal{C} is a tuple of two efficient algorithms:

- $\mathsf{Obf}(1^{\lambda}, C, \mu, \sigma) \to (\widetilde{C})$: On input 1^{λ} , a circuit $C \in \mathcal{C}_{n,m,d}(\lambda)$, a message $\mu \in \mathcal{M}_{\lambda}$ and a "lock value" $\sigma \in \{0, 1\}^{m(\lambda)}$, this algorithm outputs an obfuscated circuit \widetilde{C} .
- Eval $(\tilde{C}, x) \to \mu' \lor \bot$: On input an obfuscated circuit \tilde{C} and an input $x \in \{0, 1\}^{n(\lambda)}$, this algorithm outputs a value $\mu' \in \mathcal{M}_{\lambda}$ or \bot .

Correctness. A lockable obfuscator satisfies *(perfect) correctness* if for all $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_{n,m,d}(\lambda)$, all messages $\mu \in \mathcal{M}_{\lambda}$ and all inputs $x \in \{0,1\}^{n(\lambda)}$, the following two implications are satisfied:

1. if $C(x) = \sigma$, then $\text{Eval}(\text{Obf}(1^{\lambda}, C, \mu, \sigma), x) = \mu$ 2. if $C(x) \neq \sigma$, then $\text{Eval}(\text{Obf}(1^{\lambda}, C, \mu, \sigma), x) = \bot$

Security. We define security against multiple challenges. In [AYY22], this definition was observed to be equivalent to the original single-challenge version from [GKW17].

Definition 19 (Security against Multiple Queries). For a lockable obfuscation scheme LObf = (Obf, Eval) and an efficient algorithm Sim, we define the following oracles:

 $\begin{array}{l} \mathcal{O}\mathsf{Obf}^0(C,\mu) \text{:} \ sample \ \sigma \stackrel{*}{\leftarrow} \{0,1\}^{m(\lambda)} \ and \ return \ \widetilde{C} \leftarrow \mathsf{Obf}(1^\lambda,C,\mu,\sigma) \\ \mathcal{O}\mathsf{Obf}^1(C,\mu) \text{:} \ return \ \mathsf{Sim}(1^\lambda,1^{|C|},1^{|\mu|}) \end{array}$

We call LObf secure if there exists a PPT simulator Sim such that for all PPT adversaries \mathcal{A} , there exists a negligible function negl(\cdot) such that

$$\mathbf{Adv}_{\mathsf{LObf},\mathcal{A}}^{\mathsf{lock}}(\lambda) \coloneqq \left| \Pr\left[\mathcal{A}^{\mathcal{O}\mathsf{Obf}^1} \to 1 \right] - \Pr\left[\mathcal{A}^{\mathcal{O}\mathsf{Obf}^0} \to 1 \right] \right| \le \operatorname{negl}(\lambda)$$

Perfectly correct lockable obfuscators for general circuits are known to exist under the LWE assumption [GKW17, GKVW20].

2.8 Pseudorandom Functions (PRF)

Definition 20 (Family of Pseudorandom Functions (PRF)). Let $\{\mathcal{X}_{\lambda}\}_{\lambda \in \mathbb{N}}$, $\{\mathcal{Y}_{\lambda}\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{K}_{\lambda}\}_{\lambda \in \mathbb{N}}$ be sequences of sets representing domain, range and key space, respectively. Furthermore, let $\{\mathcal{R}_{\lambda}\}_{\lambda \in \mathbb{N}}$ be such that, for each $\lambda \in \mathbb{N}$, \mathcal{R}_{λ} is the set of all functions with domain \mathcal{X}_{λ} and range \mathcal{Y}_{λ} . A family of functions $\{\mathsf{PRF}_K\}_{K \in \mathcal{K}_{\lambda}}$ that consists of efficiently computable functions $\mathsf{PRF}_K : \mathcal{X}_{\lambda} \to \mathcal{Y}_{\lambda}$ is called pseudorandom if for all PPT adversaries \mathcal{A} , there exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\mathbf{Adv}_{\mathsf{PRF}_{K},\mathcal{A}}^{\mathsf{prf}}(1^{\lambda}) \coloneqq \left| \Pr[\mathcal{A}^{\mathsf{PRF}_{K}(\cdot)} = 1] - \Pr[\mathcal{A}^{R(\cdot)} = 1] \right| \le \operatorname{negl}(\lambda) ,$$

where $K \stackrel{\hspace{0.1em}\mathsf{\scriptscriptstyle\$}}{\leftarrow} \mathcal{K}_{\lambda}$ and $R \stackrel{\hspace{0.1em}\mathsf{\scriptscriptstyle\$}}{\leftarrow} \mathcal{R}_{\lambda}$.

It is well-known that PRFs can be constructed under DDH, *e.g.* the Naor-Reingold construction [NR97] or under the *Learning with Rounding* (LWR) [BPR12]. The LWR problem is shown to be as hard as LWE if the modulus and modulus-to-noise ratio are super-polynomial [BPR12, AKPW13].

2.9 Non-Interactive Key Exchange (NIKE)

Definition 21 (Non-Interactive Key Exchange (NIKE)). A NIKE scheme NIKE = (Setup, KeyGen, SharedKey) for a sequence of key spaces $\{\mathcal{K}_{\lambda}\}_{\lambda \in \mathbb{N}}$ is a tuple of three efficient algorithms defined as follows:

Setup(1^{\lambda}): On input the security parameter 1^{\lambda}, the algorithm outputs the public parameters PP.
KeyGen(PP): On input the public parameters PP, the algorithm outputs a pair (SK, PK) consisting of a secret key SK and the corresponding public key PK.

SharedKey(SK, PK'): On input a secret key SK and a (usually non-corresponding) public key PK', the algorithm deterministically outputs a shared key $K \in \mathcal{K}_{\lambda}$.

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

$$\Pr \begin{bmatrix} K_{1,2} = K_{2,1} & | \begin{array}{c} \mathsf{PP} \leftarrow \mathsf{Setup}(1^{\lambda}), \\ (\mathsf{PK}_1, \mathsf{SK}_1) \leftarrow \mathsf{KeyGen}(\mathsf{PP}), \\ (\mathsf{PK}_2, \mathsf{SK}_2) \leftarrow \mathsf{KeyGen}(\mathsf{PP}), \\ K_{1,2} \leftarrow \mathsf{SharedKey}(\mathsf{SK}_1, \mathsf{PK}_2), \\ K_{2,1} \leftarrow \mathsf{SharedKey}(\mathsf{SK}_2, \mathsf{PK}_1) \end{bmatrix} = 1 \ ,$$

where the probability is taken over the random coins of the algorithms.

Security. We define IND-security.

Definition 22 (IND-Security). For a NIKE scheme NIKE and a PPT adversary \mathcal{A} we define the experiment $\operatorname{Exp}_{\mathsf{NIKE},\mathcal{A}}^{\mathsf{nike},b}$ as shown in Figure 4. The oracles \mathcal{O} HonestGen, \mathcal{O} Reveal, \mathcal{O} Test and \mathcal{O} Corrupt can be called in any order and any number of times. The adversary \mathcal{A} is NOT admissible, denoted by $\operatorname{adm}(\mathcal{A}) = 0$, if either one of the following holds:

- 1. There exist public keys PK_1 and PK_2 such that \mathcal{A} made the following queries
 - \mathcal{O} Corrupt(PK₁),
 - $\mathcal{O}\mathsf{Test}(\mathsf{PK}_1,\mathsf{PK}_2)$ or $\mathcal{O}\mathsf{Test}(\mathsf{PK}_2,\mathsf{PK}_1)$ '
- 2. There exist public keys PK_1 and PK_2 such that \mathcal{A} made the following queries
 - \mathcal{O} Reveal(PK₁, PK₂) or \mathcal{O} Reveal(PK₂, PK₁),
 - $\mathcal{O}\mathsf{Test}(\mathsf{PK}_1,\mathsf{PK}_2)$ or $\mathcal{O}\mathsf{Test}(\mathsf{PK}_2,\mathsf{PK}_1)$.

Otherwise, we say that \mathcal{A} is admissible and write $\operatorname{adm}(\mathcal{A}) = 1$. We call NIKE IND-secure if for all PPT adversaries \mathcal{A} ,

$$\mathbf{Exp}_{\mathsf{NIKE},\mathcal{A}}^{\mathsf{nike}-0}(1^{\lambda}) \approx_{c} \mathbf{Exp}_{\mathsf{NIKE},\mathcal{A}}^{\mathsf{nike}-1}(1^{\lambda})$$
.

NIKE can be constructed based on a variant of the Decisional Bilinear Diffie-Hellman assumption in the standard model [FHKP13, Section 4.3]. In a recent work [Lan23], it is shown that NIKE can be constructed from LWE with polynomial modulus-to-noise ratio and satisfy strong security properties in the standard model.

$Initialize(1^{\lambda})$:	$\mathcal{O}Test(PK_1,PK_2)$:
$PP \leftarrow Setup(1^{\lambda}); \mathcal{H} \leftarrow \varnothing$	If $\{(SK_1,PK_1),(SK_2,PK_2)\} \nsubseteq \mathcal{H},$
Return PP	return \perp
	If $b = 0$, return $K \stackrel{\hspace{0.1em} \bullet}{\leftarrow} \mathcal{K}$
\mathcal{O} HonestGen():	Else, return $K \leftarrow SharedKey(SK_1, PK_2)$
$\overline{(SK,PK)} \leftarrow KeyGen$	
$\mathcal{H} \leftarrow \mathcal{H} \cup \{(SK, PK)\}$	$\mathcal{O}Corrupt(PK)$:
Return PK	Recover SK s.t. $(SK, PK) \in \mathcal{H}$
	$\mathcal{H} \leftarrow \mathcal{H} \setminus \{(SK,PK)\}$
$\mathcal{O}Reveal(PK_1,PK_2)$:	Return SK
$\overline{\text{If } \exists SK_1 \text{ s.t. } (SK_1,PK_1)} \in \mathcal{H},$	
return $K \leftarrow SharedKey(SK_1, PK_2)$	Finalize(b'):
If $\exists SK_2 \text{ s.t. } (SK_2, PK_2) \in \mathcal{H}$,	If $adm(\mathcal{A}) = 1$, return $\beta \leftarrow (b' \stackrel{?}{=} b)$
return $K \leftarrow SharedKey(SK_2, PK_1)$	Else, return $\beta \stackrel{\hspace{0.1em} {\scriptscriptstyle\bullet}}{\leftarrow} \{0,1\}$
Return \perp	

Fig. 4: Security game $\mathbf{Exp}_{\mathsf{NIKE},\mathcal{A}}^{\mathsf{nike},b}$ for $b \in \{0,1\}$

3 Technical Overview

3.1 From DMCFE to DDFE

We give an overview of our compiler from DMCFE to DDFE. The compiler is inspired by a blueprint in the literature [AGT21b] that is applied mainly in the *function-hiding* setting. The construction of [AGT21b] proceeds in two steps. First, the authors build an FH-MCFE scheme. Then, they lift their FH-MCFE scheme to FH-DDFE in a non-black-box manner. The final FH-DDFE scheme of [AGT21b] is secure against selective adversaries that submit all oracle queries up front. There is a further constraint on the adversary termed *one key-label restriction*: their queries \mathcal{O} KeyGen $(i, (k_{\text{pri}}^0, k_{\text{pub}}), (k_{\text{pri}}^1, k_{\text{pub}}))$ for $k_{\text{pub}} = (\star, \mathcal{U}_K, \text{tag-f})$ are made only *once* for each $(i, \mathcal{U}_k, \text{tag-f}), i.e.$ they do not allow repetitions for function tags. Our starting point is to aim at a transformation from DMCFE to DDFE, not from the multi-client FE regime as in [AGT21b]. The goal is to cover settings with or without function-hiding, and to achieve adaptive security for both encryption and key-generation queries under static corruption, against repetitions on message/key tags.

We explain at high level in the following paragraphs: (i) why we start from DMCFE, which differs from the work of [AGT21b], (ii) the structural properties that allow us to convert DMCFE into DDFE in a black-box manner, and (iii) a summary of the instantiations of our generic conversion on existing DMCFE schemes in the literature, as well as a new instantiation that we provide. As already presented in Table 1, we are able to cover a more general setting that applies to various schemes in the literature and achieve several DDFE constructions with previously unattained properties.

DMCFE as Starting Point - Removing the One Key-Label Restriction. We recall from the syntax and security definitions of DDFE (Definitions 4 and 5, respectively) that the key-generation $\text{KeyGen}(SK_i, k_i) \rightarrow \text{DK}_i$ and encryption $\text{Enc}(SK_i, m_i) \rightarrow \text{CT}_i$ algorithms can be seen as symmetric operations on the *i*-th slot of the key and ciphertext vectors, respectively. Each user *i* can perform these operations independently. On the other hand, when viewing MCFE as a restricted particular case of DDFE (centralized setup and key generation, fixed number of users), in order to base the DDFE scheme on an MCFE scheme, each key tag that is queried to \mathcal{O} KeyGen by the adversary must encompass a well-formed MCFE key. More specifically, for each tag-f, what the adversary is allowed to submit to \mathcal{O} KeyGen must only correspond to a *global* key query on $(k_i)_{i \in [n]}$ of all n

components, where n is a fixed number of users and each $i \in [n]$ is assigned one k_i . Translating back to the language of DDFE, this implies for each $(i, \mathsf{tag-f})$ the adversary is allowed to submit only one query $\mathcal{O}\mathsf{KeyGen}(i, (k_{\mathrm{pri}}^0, k_{\mathrm{pub}}), (k_{\mathrm{pri}}^1, k_{\mathrm{pub}}))$, which leads exactly to the one key-label restriction. As a consequence, in order to circumvent this restriction it is necessary to start from a generalization of MCFE that allows for *local* key queries, which is the DMCFE model. Then, the next step is to devise a transformation from DMCFE to DDFE that is not subject to the one key-label restriction, *i.e.* a transformation that allows for repeated key queries for a slot *i* under the same key tag. By doing so, starting from a DMCFE that is secure against repetitions on key tags, our obtained DDFE scheme inherits the stronger security notion compared to [AGT21b].

Transformation from (D)MCFE to DDFE. As we have mentioned in Section 1, the striking difference is that in DDFE there is only a non-interactive global setup outputting public parameters. So as to join the system, each user runs a *local* setup algorithm to generate their own secret key SK_i using some public parameters. At any time, any set of users \mathcal{U}_M can independently encrypt their individual data to contribute to a list of ciphertexts $(\mathsf{CT}_i)_{i \in \mathcal{U}_M}$ under some message tag tag. Similarly, a set of users \mathcal{U}_K can independently contribute to a list of functional keys $(\mathsf{DK}_i)_{i \in \mathcal{U}_K}$ under some key tag tag-f. Note that the dynamic set of users is also reflected in the DDFE version of the function class for inner products (cf. Definition 6), and for the function class of AB-AWS (cf. Definition 12 for inner products and Definition 13 for AB-AWS). Hence, it is necessary to deal with newly arrived sets of users if one wants to convert a DMCFE scheme to a DDFE scheme, *i.e.* to handle in an independent manner each \mathcal{U}_M (for encryption) and \mathcal{U}_K (for key-generation) by the underlying DMCFE.

From Previous Works - The Transformation of [AGT21b]. We briefly highlight the key ideas of the transformation from MCFE to DDFE in [AGT21b]. Let ID be some set of identities. The authors of [AGT21b] start from an MCFE in which each user $i \in [n] \subseteq ID$ holds a master secret key MSK_i of a single-input IPFE scheme. First of all, the removal of interaction between the users, for the sake of obtaining their SK_i, is achieved by equipping each user *i* with a key K_i for a family of pseudorandom functions $\{PRF_K\}_K$. Then, for each independent support $\mathcal{U}_M \subseteq ID$, user $i \in \mathcal{U}_M$ runs the setup algorithm of the single-input FE using $PRF_{K_i}(\mathcal{U}_M) \to r_i$ as fixed random coins. The PRF ensures that for the same support \mathcal{U}_M , user *i* always uses the same key SK_i, but different independent keys for different supports. Next, concerning the key-generation, the authors of [AGT21b] use a well-known technique from the literature $[CDSG^+20]$ called *Decentralized Sum* (DSum). From a bird's eye view, DSum is used to generate a fresh secret sharing of 0 for each function, by interleaving a Non-Interactive Key Exchange (NIKE) scheme and a PRF, without interaction and independently for each support \mathcal{U}_K . This addresses the challenge of key-generation in a decentralized and non-interactive manner when moving to DDFE.

<u>Our Conversion from</u> Dynamizable <u>DMCFE to DDFE</u>. We observe that both techniques mentioned in the previous paragraph can be applied in a much broader setting. As mentioned above, our starting point is a DMCFE scheme. Intuitively, we show that a DMCFE scheme can be lifted to DDFE whenever the only correlation between the user's secret keys comes from a random secret sharing of 0. Specifically, we require that the *i*-th secret key SK_i must be possible to generate given only some global public parameters and the *i*-th share of an *n*-out-of-*n* secret sharing of 0, where 0 is the neutral element of an arbitrary finite Abelian group. This property is coined dynamizability of the DMCFE scheme and we give the formal definition in Definition 23. As in [AGT21b], we use a DSum instance to compute a secret sharing $(s_i)_i$ without interaction and a PRF to generate the key components independent of other users. In this way, we can emulate an independent DMCFE instance with respect to each support \mathcal{U}_M and \mathcal{U}_K , and each user *i* can use their DDFE secret key SK_i to dynamically derive a DMCFE key for encryption and/or key-generation in the DMCFE scheme for arbitrary sets \mathcal{U}_M and \mathcal{U}_K .

Concrete Instantiations. We first discuss the case for the function class of inner products, whose definitions are given in Definition 12 for DMCFE functionality and Definition 6 for DDFE functionality. The compiler first allows us to apply the conversion to the FH-DMCFE scheme from [NPS24], where the shares s_i are vectors in \mathbb{Z}_q^2 . (Recall that their scheme uses two scalar secret sharings, one in the keys and one in the ciphertexts). This gives us the first function-hiding IP-DDFE construction with adaptive security in the literature. Moreover, we also note that the FH-DMCFE scheme from [NPS24] achieves security against a fixed polynomially bounded number of repetitions on the key tags, meaning the resulting FH-IP-DDFE scheme inherits this property and improves upon the work of [AGT21b]. Similarly, our conversion can be applied to the DMCFE scheme in [CDG⁺18a] where shares s_i are matrices in $\mathbb{Z}_q^{2\times 2}$, giving the first adaptively secure scheme in the standard (non-function-hiding) setting. Details about these instantiations can be found in Appendices A.5 and A.3, respectively.

Furthermore, we want to apply our conversion to the lattice-based DMCFE of [LT19]. Unfortunately, their scheme does not exactly meet our definition of dynamizability. The problem is that the secret keys SK_i in their scheme do not contain a secret share of 0, but a secret share t_i of a value t distributed according to a discrete Gaussian. Nevertheless, we are able to make the scheme fit into our framework via a slight modification. Without recalling the details of their scheme here, our key observation is that encryption and key generation for some slot i can be done using only the share t_i . The sum $t = \sum_{i \in [n]} t_i$ is only needed for *decryption*. For this reason, it is not necessary to include the value t into the secret keys or the public parameters (as done in [LT19]). Instead, we can mask the value t_i with a share s_i of a secret sharing of 0 which can be computed using the DSum technique as before. This gives a variant of [LT19]. Specifically, when every user includes the masked value $u_i = t_i + s_i$ into their decryption keys (or ciphertexts), then the sum $\sum_{i \in [n]} u_i = \sum_{i \in [n]} t_i = t$ can be reconstructed at decryption time. It may seem that this already solves all our problems, but there is one more technical detail. The DSum technique works only for *finite* Abelian groups. However, the support of discrete Gaussian random variables is over \mathbb{Z} and infinite. Therefore, our argument for dynamizability, later the security of our modification to [LT19], becomes probabilistic in the sense that: with high probability, we will be working in a fixed finite range and the argument holds for statistically close distributions (in previous applications to [CDG⁺18a, NPS24] the dynamizability is perfect). In the end, we are able to apply our conversion to the lattice-based DMCFE scheme of Libert et al. [LT19] which yields an IP-DDFE whose security is based solely on LWE in the standard model. We refer to Appendix A.4 for details.

For the functionality of AB-AWS, whose definitions are given in Definition 13 for DMCFE and Definition 7 for DDFE, we construct a new DMCFE scheme in Appendix A.2. We do not recall the details of this construction here because it is very similar to a scheme for function-hiding IP-DDFE recently described in [NPS24]. The only difference is that they start from a single-client function-hiding FE scheme for inner products, whereas we use a single-client FE scheme for a functionality termed AB-AWSw/IP introduced and instantiated in [ATY23]. The resulting scheme for AB-AWS is dynamizable with shares s_i being elements of \mathbb{Z}_q . The IP-DDFE scheme in [NPS24] is only secure against so-called complete queries. This restriction in the security model can be removed via a generic conversion also discussed in [NPS24]. When considering functionalities with access control (such as AB-AWS), however, then the situation is more complex which is why the conversion of [NPS24] to achieve security against incomplete queries does not suffice anymore. To deal with this more complex situation and achieve security against any queries, we present a second compiler whose details are discussed in the next section.

3.2 Achieving Security Against Any Queries

So far, we have not paid attention to one important issue. Our DMCFE scheme for AB-AWS which has been lifted to DDFE for AB-AWS is only secure against so-called *legitimate* queries. Intuitively, a query is legitimate if it is subject to the adversary's admissibility condition. In more detail, for some set of users $\mathcal{U} \subseteq ID$ and tags tag, tag-f \in Tag, there may exist an honest slot $i \in \mathcal{U}$ such that the adversary does not have a ciphertext with respect to tag and an attribute x_i , and a decryption key with respect to tag-f and a policy f_i such that $f_i(x_i) = 0$. If this happens, the adversary is not supposed to learn anything, as the decryption algorithm cannot be run in an honest manner on this input. Therefore, the admissibility condition does not impose any restriction in this case, and we call the adversary's queries with respect to \mathcal{U} and tag, tag-f *illegitimate*.

Existing Solutions in the MIFE Setting and Their Limitations. In [ATY23], the authors show how to lift an MIFE for AB-AWS from legitimate-query to any-query security. Roughly speaking, they use an *n*-out-of-*n* secret sharing to share the decryption keys of the underlying MIFE which is secure against legitimate queries. Then they encrypt each share with an independent ABE. If a query is illegitimate, then not all ABE ciphertexts can be decrypted and the MIFE key cannot be recovered. We can observe that this compiler only works in the MIFE setting because it cannot deal with different tags. Indeed, once all shares are recovered and the MIFE decryption key is reconstructed, it can be used for a decryption of ciphertexts associated with arbitrary tags, regardless of whether they are legitimate or not. To overcome this problem, we need a more powerful primitive than ABE which is able to check legitimacy *globally* across several ciphertexts, as in the presence of tags this property cannot be decided by looking at only one slot at a time anymore.

In other words, the primitive we are looking for is *Multi-Input ABE* (MI-ABE). MI-ABE allows each encryptor $i \in [n]$ to generate a ciphertext CT_i with respect to an attribute x_i . Using a decryption key for some arity-*n* policy f, (CT_1, \ldots, CT_n) can be jointly decrypted if the combination of all attributes satisfies the key's policy, *i.e.* if $f(x_1, \ldots, x_n) = 0$. By viewing the tags as part of the attributes, MI-ABE seems powerful enough to perform a tag-sensitive check of legitimacy. Unfortunately, all existing constructions of MI-ABE are either based on nonstandard assumptions [AYY22, ARYY23] or their supported policy classes capture only conjunctions [FFMV23, ATY23]⁶. Note that conjunctions are *not* powerful enough to check a global equality, hence do not suffice to check legitimacy with respect to a specific tag⁷. So we need to construct such a scheme first. Fortunately, there is one work that comes somewhat close to what we need: in [FFMV23], Francati *et al.* build multi-input predicate encryption from any classical predicate encryption (PE) scheme [GVW15] and lockable obfuscation [GKW17]. Their supported policy class is conjunctions of \mathcal{F} , where \mathcal{F} is the class of policies supported by the employed PE scheme. Before we explain how tags can be integrated into their construction, we give a brief recall of their techniques.

Ingredients. First, Lockable Obfuscation (LO) [GKW17, WZ17] allows to obfuscate a circuit C with respect to a message μ and a lock value σ . Correctness asks that an evaluation of the obfuscated circuit on some input x yields μ if $C(x) = \sigma$ and \perp otherwise. Simulation security requires that if σ looks random to the adversary, then the obfuscated circuit is computationally indistinguishable from a garbage program that does not carry any information about μ or C. Second, Attribute-Based Encryption (ABE) enables the generation of ciphertexts $\mathsf{aCT}(x,\mu)$ for a message μ with respect to an attribute x and decryption keys $\mathsf{aDK}(f)$ with respect to a policy f. Correctness and security

⁶ A policy f is said to be a *conjunction* of a policy class \mathcal{F} if there exist policies $f_1, \ldots, f_n \in \mathcal{F}$ such that $f(x_1, \ldots, x_n) = f_1(x_1) \wedge \cdots \wedge f_n(x_n)$.

⁷ If no tag is given explicitly at decryption time, which is our setting, two main reasons for this insufficiency are: (i) message tags are not known at key-generation time, (ii) each conjunction term $f_i(x_i)$ evaluates attributes at slot i independently from $f_j(x_j)$ on attributes at slot $j \neq i$, cf. [ACGU20, NPP22] and discussions therein for details.

require that decryption is possible if and only if f(x) = 0. In the particular case of [FFMV23], attributes are of the form $x = (x_1, \ldots, x_n)$ and policies are of the form $f(x) = f_1(x_1) \wedge \cdots \wedge f_n(x_n)$. For each $i \in [n]$, there exists a wildcard \star such that $f_i(\star) = 0$ for any choice of f_i . Similarly, we consider identity-based encryption (IBE) which allows the generation of ciphertexts idCT(id, μ) with respect to an identity id and a message μ , and decryption keys idDK(id') with respect to an identity id'. Decryption is possible if and only if id = id'. Furthermore, we use several independent PKE instances. An encryption of a message μ under the public key of the *i*-th instance is denoted by $pCT_i(\mu)$.

The MI-ABE of [FFMV23]. For simplicity, we consider a single-message scheme where the first slot takes an attribute and a message whereas the other slots take only an attribute⁸. Furthermore, we do not consider attribute-hiding in this work, which allows us to start from any ABE instead of PE. Each user $i \in [n]$ in the MI-ABE holds the secret key pSK_i of an independent PKE instance. To generate a ciphertext CT_{x_i} for an attribute x_i (and a message μ if i = 1), user i samples a random lock value σ_i and creates $c_i^{(0)} = \mathsf{aCT}((x_1, \ldots, x_n), \sigma_i)$, where $x_j = \star$ for all $j \neq i$. Subsequently, the encryptor adds n layers of PKE encryption: for all $j = 1, \ldots, n$, they compute $c_i^{(j)} = \mathsf{pCT}_j(c_i^{(j-1)})$. The final ciphertext is an obfuscation $\mathsf{CT}_{x_i} = \widetilde{C}_i$ of a circuit $C_i[c_i^{(n)}, \mathsf{pSK}_i]$ generated with respect to the lock value σ_i and the message μ (if i = 1) or the PKE key pSK_i (if i > 1). The notation $C[\alpha]$ means that the value α is hardwired in the description of the circuit C. MI-ABE decryption keys DK_f for a policy f are simply a decryption key $\mathsf{aDK}(f)$ of the employed single-input ABE.

The pivotal point of the construction is the definition of the circuits $C_i[c_i^{(n)}, \mathsf{pSK}_i]$ which must enable "communication" between the obfuscated circuits without violating attribute privacy. To recover the message μ , we must unlock C_1 . The corresponding lock value σ_1 is already hardwired in the circuit $C_1[c_1^{(n)}, \mathsf{pSK}_1]$, however it is hidden under n layers of PKE encryption in the value $c_1^{(n)}$. To decrypt, we need the secret keys pSK_i for all i > 1. These keys are embedded as messages in the obfuscated circuits \tilde{C}_i . So in order to evaluate \tilde{C}_1 , we need to evaluate \tilde{C}_i for i > 1 first. Specifically, $C_1[c_1^{(n)}, \mathsf{pSK}_1]$ takes as input $\mathsf{aDK}(f)$ and invokes the evaluation of \widetilde{C}_2 on input $(\mathsf{pSK}_1, \mathsf{aDK}(f))$, \widetilde{C}_2 invokes the evaluation \widetilde{C}_3 on input $(\mathsf{pSK}_1, \mathsf{pSK}_2, \mathsf{aDK}(f))$ and so on until \widetilde{C}_n takes the secret keys from all the other slots as input. At this point, the evaluation of C_n , which is an obfuscation of $C_n[c_n^{(n)}, \mathsf{pSK}_n]$, has everything to remove the *n* layers of PKE encryption from $c_n^{(n)}$ to recover $c_n^{(0)} =$ $\operatorname{aCT}((x_1,\ldots,x_n),\sigma_n)$. Recall that $x_1 = \cdots = x_{n-1} = \star$ in $c_n^{(0)}$. Thus, if $f_n(x_n) = 0$, then the evaluation of \widetilde{C}_n can further decrypt $c_n^{(0)}$ using $\mathsf{aDK}(f)$ to recover σ_n which unlocks \widetilde{C}_n and reveals pSK_n . Now \widetilde{C}_{n-1} has pSK_n from the evaluation of \widetilde{C}_n , pSK_{n-1} hardwired in its own description, and $(\mathsf{pSK}_1, \ldots, \mathsf{pSK}_{n-2}, \mathsf{aDK}(f))$ from its inputs. So \widetilde{C}_{n-1} can perform a similar computation as \widetilde{C}_n at the end of which the secret key pSK_{n-1} is revealed and so on. Eventually, \widetilde{C}_1 recovers all the secret keys from the other slots via nested evaluations of the other obfuscated circuits, which allows recovering σ_1 and unlocking C_1 to learn μ . (The actual decryption procedure is a bit more complex leading to a runtime of $O(n^n)$ which limits the number of slots to n = O(1), but we omit the details here).

Dealing With Tags. We first observe that the policies supported by the construction of [FFMV23] are not able to perform a global equality check. This is because in the generation of a ciphertext \widetilde{C}_i all attributes x_j , for $j \in [n] \setminus \{i\}$, are set to the wildcard \star . Hence, even if one could generate keys for arbitrarily powerful policies, it is not possible to check authorization with respect to attributes from more than one slot at the same time. We therefore need to approach the problem differently, without embedding the condition on tag equality into the policies. For this, we recall that decryption uses a sophisticated nesting technique of several obfuscated circuits which only works if

⁸ Such a single-message scheme can easily be lifted to an *n*-message scheme by running multiple single-message schemes in parallel with rotated slots [AYY22].

the circuits are linked properly. This "link" is established as follows. To unlock an obfuscation \tilde{C}_i of a circuit $C_i[c_i^{(n)}, \mathsf{pSK}_n]$, one needs to recover the lock value σ_i . This lock value is hardwired in the circuit itself, but it is hidden in $c_i^{(n)}$ under n layers of PKE encryption. The corresponding secret keys pSK_j are embedded in the obfuscated circuits \tilde{C}_j from the other slots $j \in [n]$; so to recover σ_i , the PKE keys from the other slots must be retrieved first⁹. Our idea is to let the n encryption layers in $c_i^{(n)}$ depend on the tag. Specifically, the j-th layer $c_i^{(j)}$ should only be possible to decrypt if the corresponding secret key, which is embedded in \tilde{C}_j , is associated with the same tag. In this way, it can be checked in a pairwise manner that \tilde{C}_i and \tilde{C}_j were generated with respect to the same tag tag. To this end, we replace the PKE instances with IBE. During the generation of \tilde{C}_i , the value $c_i^{(j)}$, for $j \in [n]$, is generated by encrypting $c_i^{(j-1)}$ under the public key of the j-th IBE instance with respect to the tag tag which is viewed as the identity, *i.e.* $c_i^{(j)} = \mathsf{idCT}_j(\mathsf{tag}, c_i^{(j-1)})$. Correspondingly, the above-mentioned link between two obfuscated circuits works only if both were generated with respect to the same tag.

From MI-ABE to DDFE. Our next step is to lift this construction from the MIFE setting to DDFE. On the positive side, the initial construction from [FFMV23] is already secure under corruptions, and we have already discussed how to encrypt with respect to tags. It remains to modify the generation of decryption keys so that it works in a decentralized manner and does not require a master secret key anymore. Furthermore, we need to allow users to join the system dynamically. This requires to modify the setup algorithm so that it works without interaction. To our knowledge, there is currently no work in the literature that considers (key-policy) ABE in such a general setting. We therefore start by introducing a new functionality that we call *Attribute-Based All-or-Nothing Encapsulation* (AB-AoNE), as it can be regarded as a generalization of the AoNE functionality from [CDSG⁺20]. Consider a tag tag and a set of users \mathcal{U} . Given a ciphertext of a message μ_i created with respect to \mathcal{U} , tag and an attribute x_i and a decryption key generated with respect to \mathcal{U} and a policy f_i for each $i \in \mathcal{U}$, the AB-AoNE functionality allows recovering $\{\mu_i\}_{i\in\mathcal{U}}$ if and only if $f_i(x_i) = 0$ for all $i \in \mathcal{U}$.

We first discuss how to modify the construction such that key generation works in a decentralized manner. Previously, to generate a key for a policy f, one runs the key generation algorithm of the employed ABE scheme on input f. This requires the ABE master secret key and, hence, can only be performed by a central authority that cannot be corrupted. To overcome this limitation, we exploit two properties. First, we recall that policies f are always of the form $f = f_1 \land \cdots \land f_n$. Second, we observe that ciphertexts are always generated with respect to attributes (x_1, \ldots, x_n) where all but one x_i are wildcards. Together, these two properties imply that the ABE is solely used to perform *local* checks of authorization in one slot, but never a global check across several slots at the same time. For this reason, we can replace the global ABE instance with independent instances, where each user i holds the ABE master secret key of the i-th instance and generates a partial decryption key $\mathsf{DK}_i = \mathsf{a}\mathsf{DK}_i(f_i)$.

Second, the DDFE model asks that functions can be evaluated with respect to dynamically chosen subsets \mathcal{U} of users. Specifically, we must guarantee decryptability for ciphertexts and decryption keys generated for the same set \mathcal{U} of users, whereas we must prevent the adversary from meaningfully combining ciphertexts and keys generated with respect to different sets $\mathcal{U}' \neq \mathcal{U}$. To achieve this, our idea is to again manipulate the "link" between the obfuscated circuits \tilde{C}_i . Recall that \tilde{C}_i is an obfuscation of a circuit $C_i[c_i^{(n)}, idDK(tag)]$, where $c_i^{(n)}$ is generated by a single ABE encryption

⁹ More precisely, if the evaluation of \tilde{C}_i is invoked by an obfuscated circuit \tilde{C}_j with j < i, then the PKE secret key pSK_j is directly given as input. Conversely, to recover the secret key pSK_j for j > i, \tilde{C}_i invokes \tilde{C}_j by itself and pSK_j is released if the evaluation succeeds.

followed by n layers of IBE encryption. For the ABE, the situation is simple as both encryption and key generation are performed by the same user. Therefore, we can simulate an independent ABE instance for each set \mathcal{U} by using a PRF for the generation of the random coins passed to the ABE setup algorithm. For the IBE, the situation is slightly different. This is because during encryption with respect to some set \mathcal{U} , a user i must compute $c_i^{(n)}$ which requires the IBE public keys of the other users $j \in \mathcal{U} \setminus \{i\}$. As encryption should be possible without interaction, we cannot distribute fresh IBE public keys for each new set \mathcal{U} . Instead, users must provide a fixed public key when they join the system. These public keys cannot depend on specific sets \mathcal{U} since they are not known at this point yet. To overcome this problem, we exploit again the fact that we use IBE instead of PKE which allows us to embed the current set \mathcal{U} as part of the identities.

DDFE for AB-AWS Secure Against Any Queries. Being equipped with a DDFE for AB-AoNE, we can tackle our original task of building a DDFE for AB-AWS secure against any queries. Let wmFE = (wmGSetup, wmLSetup, wmKeyGen, wmEnc, wmDec) be a DDFE for AB-AWS secure against legitimate queries and anFE = (anGSetup, anLSetup, anKeyGen, anEnc, anDec) a DDFE for AB-AONE. We consider a polynomial-size function tag space Tag-f. To set up a user *i*, our final DDFE generates an independent instance of wmFE and anFE for each tag-f \in Tag-f. To encrypt $m_i = (m_{i,pri}, m_{i,pub})$ with $m_{i,pub} = (\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_j, \mathcal{U}_M, \mathsf{tag})$, one runs wmCT_{*i*,tag-f} \leftarrow wmEnc(wmSK_{*i*,tag-f}, m_i) and outputs CT_{*i*,tag-f} \leftarrow anEnc(anSK_{*i*,tag-f}, (wmCT, $\mathcal{U}_M, \mathsf{tag}$)) for each tag-f. Key generation for an input $k_i = (g_i, h_i, \mathcal{U}_K, \mathsf{tag-f'})$ runs and outputs wmDK_{*i*,tag-f'} \leftarrow wmKeyGen(wmSK_{*i*,tag-f}, k_i) and anDK_{*i*,tag-f} \leftarrow anKeyGen(anSK_{*i*,tag-f}, \mathcal{U}_K). If one has an authorized combination of ciphertexts and decryption keys, then one can first remove the layer of AB-AoNE encryption, followed by the decryption of the scheme secure against legitimate queries. For security, if the adversary submits an illegitimate query, then the AB-AoNE ciphertext cannot be decrypted. Otherwise, if the query is legitimate, then the AB-AoNE can be removed but we can rely on the security of the original scheme wmFE.

Open Problems - Decryption Efficiency and Larger Key Tag Space. Our final DDFE has two limitations. First, it can only support *constant-size sets* \mathcal{U} *of users.* This limitation is inherited from the MI-ABE of [FFMV23] whose decryption procedure is only efficient for a constant number of slots. Second, our scheme can only deal with a function tag space of polynomial size. One reason for this limitation is that the key generation of our AB-AoNE functionality cannot deal with tags. This is again related to the original scheme from [FFMV23] which is not secure under *collusions.* However, even if we were able to generalize the AB-AoNE construction to function tags, it is unclear how this would help to build DDFE for AB-AWS for a larger function tag space due to mix-and-match attacks. Indeed, having one legitimate combination with decryption keys for some tag tag-f suffices to remove the AB-AoNE layer and recover the wmFE ciphertexts. Subsequently, these wmFE ciphertexts can be decrypted by any (even illegitimate) keys for another tag tag-f'. To prevent these attacks, our above compiler uses an independent wmFE instance for each tag-f, but this fact also limits the function tags to a polynomial number.

Finally, we would like to emphasize that solving any of these problems is not only interesting for our current task of achieving DDFE for AB-AWS with security against any queries, but would also have implications to future constructions of MI-ABE.

4 From DMCFE to DDFE

In this section, we generically build DDFE schemes from DMCFE, PRF and NIKE schemes. The setup algorithms of the underlying DMCFE schemes are required to satisfy simple structural properties.

Definition 23 (Dynamizability). Let \mathbb{A} be an (additively written) finite Abelian group and $n \in \mathbb{N}$. We define the set of n-out-of-n sharings of $0_{\mathbb{A}}$ as $\mathcal{S}(n, \mathbb{A}) = \{(s_i)_{i \in [n]} \in \mathbb{A}^n : \sum_{i \in [n]} s_i = 0_{\mathbb{A}}\}.$

A DMCFE scheme FE = (Setup, KeyGen, Enc, Dec) is called A-dynamizable if there exist PPT algorithms SetupPP and SetupUser such that, for all $\lambda, n \in \mathbb{N}$, the distributions

$$\left\{ \mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]} \middle| \begin{array}{c} (s_i)_{i \in [n]} \xleftarrow{\hspace{1.5pt} \$} \mathcal{S}(n, \mathbb{A}) \\ \mathsf{PP} \leftarrow \mathsf{Setup}\mathsf{PP}(1^{\lambda}) \\ \forall i \in [n] \colon \mathsf{SK}_i \leftarrow \mathsf{Setup}\mathsf{User}(\mathsf{PP}, s_i) \end{array} \right\}$$

and $\{(\mathsf{PP}, \{\mathsf{SK}_i\}_{i\in[n]}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^n)\}$ are identical, where the probability is taken over the sampling of $(s_i)_{i\in[n]}$ and the random coins of the algorithms.

Next, we define the DDFE functionality f^{dyn} that we will obtain when plugging a DMCFE scheme for a functionality f into our conversion.

Definition 24 (Corresponding DDFE Functionality). Let $\{\mathsf{Tag}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathsf{ID}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathcal{K}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathcal{M}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathcal{M}_{\lambda}\}_{\lambda\in\mathbb{N}}, \{\mathcal{M}_{\lambda}\}_{\lambda\in\mathbb{N}} \text{ and } \{\mathcal{R}_{\lambda}\}_{\lambda\in\mathbb{N}} \text{ be sequences of tag, identity, key, message and output spaces, respectively, where <math>\mathsf{Tag}_{\lambda} = \mathsf{ID}_{\lambda} = \{0,1\}^{\mathsf{poly}(\lambda)}$ and $\mathcal{K}_{\lambda} = \mathcal{K}_{\lambda,\mathsf{pri}} \times \mathcal{K}_{\lambda,\mathsf{pub}}, \mathcal{M}_{\lambda} = \mathcal{M}_{\lambda,\mathsf{pri}} \times \mathcal{M}_{\lambda,\mathsf{pub}}$ consist of a private and a public component each. Furthermore, let $\mathcal{K}_{\lambda,\mathsf{pub}}^{\mathsf{dyn}} = \mathcal{K}_{\lambda,\mathsf{pub}} \times 2^{\mathsf{ID}_{\lambda}} \times \mathsf{Tag}_{\lambda}$ and $\mathcal{M}_{\lambda,\mathsf{pub}}^{\mathsf{dyn}} = \mathcal{M}_{\lambda,\mathsf{pub}} \times 2^{\mathsf{ID}_{\lambda}} \times \mathsf{Tag}_{\lambda}$, then define $\mathcal{K}_{\lambda}^{\mathsf{dyn}} = \mathcal{K}_{\lambda,\mathsf{pri}} \times \mathcal{K}_{\lambda,\mathsf{pub}}^{\mathsf{dyn}}$ and $\mathcal{M}_{\lambda} = \mathcal{M}_{\lambda,\mathsf{pri}} \times \mathcal{M}_{\lambda,\mathsf{pub}}^{\mathsf{dyn}}$. Consider a DMCFE functionality $f = \{f_{\lambda,n}: \mathcal{K}_{\lambda}^n \times \mathcal{M}_{\lambda}^n \to \mathcal{R}_{\lambda}\}_{\lambda,n\in\mathbb{N}}$. The DDFE functionality

$$f^{\mathsf{dyn}} = \left\{ f_{\lambda}^{\mathsf{dyn}} \colon \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_{\lambda} \times \mathcal{K}_{\lambda}^{\mathsf{dyn}})^n \times \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_{\lambda} \times \mathcal{M}_{\lambda}^{\mathsf{dyn}})^n \to \mathcal{R}_{\lambda} \right\}_{\lambda \in \mathbb{N}}$$

which corresponds to the DMCFE functionality f is defined via

$$f_{\lambda}^{\mathsf{dyn}}(\{i, k_i^{\mathsf{dyn}}\}_{i \in \mathcal{U}_K}, \{i, m_i^{\mathsf{dyn}}\}_{i \in \mathcal{U}_M}) = \begin{cases} f_{\lambda, |\mathcal{U}|}(\{k_i\}_{i \in \mathcal{U}}, \{m_i\}_{i \in \mathcal{U}}) & \text{if } (*) \text{ holds} \\ \bot & \text{otherwise} \end{cases}$$

for every $\lambda \in \mathbb{N}$, where condition (*) holds if $\mathcal{U}_K = \mathcal{U}_M$ (in which case we define $\mathcal{U} \coloneqq \mathcal{U}_K$) and there exist tag, tag-f $\in \mathsf{Tag}_\lambda$ such that for each $i \in \mathcal{U}$, we require (i) k_i^{dyn} is of the form $(k_{i,\mathrm{pri}}, (k_{i,\mathrm{pub}}, \mathcal{U}, \mathsf{tag-f}))$ (in which case we define $k_i \coloneqq (k_{i,\mathrm{pri}}, k_{i,\mathrm{pub}})$), and (ii) m_i^{dyn} is of the form $(m_{i,\mathrm{pri}}, (m_{i,\mathrm{pub}}, \mathcal{U}, \mathsf{tag}))$ (in which case we define $m_i \coloneqq (m_{i,\mathrm{pri}}, m_{i,\mathrm{pub}})$).

For example, we can observe that the DDFE inner-product functionality f^{dyn-ip} in Definition 6 corresponds to the DMCFE inner-product functionality f^{ip} in Definition 12, and that the DDFE attribute-based attribute-weighted sums functionality $f^{dyn-ab-aws}$ in Definition 7 corresponds to the DMCFE version f^{ab-aws} in Definition 13. We next describe our conversion.

Construction 25 (DMCFE to DDFE). The construction uses the following ingredients:

- An A-dynamizable DMCFE scheme mFE = (mSetup, mKeyGen, mEnc, mDec) for a functionality f. As mFE is A-dynamizable, it is equipped with two additional algorithms mSetupPP and mSetupUser.
- Two families of pseudorandom functions $\{\mathsf{PRF}_K\}_{K\in\mathcal{K}}$ and $\{\mathsf{PRF}'_K\}_{K\in\mathcal{K}'}$, where the range of $\{\mathsf{PRF}'_K\}_{K\in\mathcal{K}'}$ is a subset of \mathbb{A} .
- A NIKE scheme NIKE = (nSetup, nKeyGen, nSharedKey) with key space \mathcal{K}' .

The details of our DDFE scheme FE = (GSetup, LSetup, KeyGen, Enc, Dec) for the functionality f^{dyn} corresponding to f are as follows:

- GSetup (1^{λ}) : On input the security parameter 1^{λ} , run mPP \leftarrow mSetupPP (1^{λ}) and nPP \leftarrow nSetup (1^{λ}) and output PP := (mPP, nPP)
- LSetup(*i*): On input an identity $i \in ID$, sample $K_i \stackrel{s}{\leftarrow} \mathcal{K}$, generate $(nSK_i, nPK_i) \leftarrow nKeyGen(nPP)$ and output the key pair $(PK_i \coloneqq nPK_i, SK_i \coloneqq (nSK_i, K_i))$.
- $\mathsf{KeyGen}(\mathsf{SK}_i, k_i^{\mathsf{dyn}}): \text{ On input a secret key } \mathsf{SK}_i = (\mathsf{nSK}_i, K_i) \text{ and } k_i = (k_{i, \mathrm{pri}}, k_{i, \mathrm{pub}} = (k'_{i, \mathrm{pub}}, \mathcal{U}_K, \mathsf{tag-f}))$ such that $i \in \mathcal{U}_K$, compute and output DK_i as follows: $\forall j \in \mathcal{U}_K \setminus \{i\}$:

$$\begin{split} K'_{i,j} &\leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_i,\mathsf{nPK}_j); \ s_i = \sum_{j \in \mathcal{U}_K \setminus \{i\}} (-1)^{j < i} \mathsf{PRF}'_{K'_{i,j}}(\mathcal{U}_K) \\ \mathsf{mSK}_i &\leftarrow \mathsf{mSetupUser}(\mathsf{mPP}, s_i; \mathsf{PRF}_{K_i}(\mathcal{U}_K)) \\ \mathsf{DK}_i &\leftarrow \mathsf{mKeyGen}(\mathsf{mSK}_i, \mathsf{tag-f}, (k_{i, \mathrm{pri}}, k'_{i, \mathrm{pub}})) \end{split}$$

Enc(SK_i, m_i^{dyn}): On input a secret key SK_i = (nSK_i, K_i) and $m_i = (m_{i,pri}, m_{i,pub} = (m'_{i,pub}, \mathcal{U}_M, tag))$ such that $i \in \mathcal{U}_M$, compute and output CT_i as follows: $\forall j \in \mathcal{U}_M \setminus \{i\}$:

$$\begin{split} & K'_{i,j} \leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_i,\mathsf{nPK}_j); \ s_i = \sum_{j \in \mathcal{U}_M \setminus \{i\}} (-1)^{j < i} \mathsf{PRF}'_{K'_{i,j}}(\mathcal{U}_M) \\ & \mathsf{mSK}_i \leftarrow \mathsf{mSetupUser}(\mathsf{mPP},s_i;\mathsf{PRF}_{K_i}(\mathcal{U}_M)) \\ & \mathsf{CT}_i \leftarrow \mathsf{mEnc}(\mathsf{mSK}_i,\mathsf{tag},(m_{i,\mathrm{pri}},m'_{i,\mathrm{pub}})) \end{split}$$

 $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in\mathcal{U}_K},\{\mathsf{CT}_i\}_{i\in\mathcal{U}_M})$: On input a set of decryption keys $\{\mathsf{DK}_i\}_{i\in\mathcal{U}_K}$ and a set of ciphertexts $\{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}$, if $\mathcal{U}_K \neq \mathcal{U}_M$ abort with failure, otherwise compute and output

$$d \leftarrow \mathsf{mDec}(\{\mathsf{DK}_i\}_{i \in \mathcal{U}_K}, \{\mathsf{CT}_i\}_{i \in \mathcal{U}_M})$$
.

Correctness It holds that $\sum_{i \in \mathcal{U}_K} s_i = \sum_{i \in \mathcal{U}_K} \sum_{j \in \mathcal{U}_K \setminus \{i\}} (-1)^{j < i} \mathsf{PRF}'_{K'_{i,j}}(\mathcal{U}_K)$ and evaluates to 0 thanks to the correctness of the NIKE scheme NIKE that gives $K'_{i,j} = K'_{j,i}$ for all $i, j \in \mathcal{U}_K$. Thus, $(s_i)_{i \in \mathcal{U}_K} \in \mathcal{S}(|\mathcal{U}_K|, \mathbb{A})$. The argument for \mathcal{U}_M proceeds in the same way. Then the correctness of FE follows from the correctness of mFE and the decomposition of the setup algorithm according to the \mathbb{A} -dynamizability.

Security Security is stated in the following proposition.

Proposition 26. Let $yyy \in \{sel, sadap, adap\}, zzz \in \{sym, asym\}$. If mFE is stat-yyy-zzz-secure and \mathbb{A} -dynamizable, $\{PRF_K\}_{K \in \mathcal{K}}$ and $\{PRF'_{K'}\}_{K' \in \mathcal{K}'}$ are pseudorandom and NIKE is secure, then the DDFE scheme FE in Construction 25 is also stat-yyy-zzz-partially function-hiding. Moreover, if mFE is secure without repetitions¹⁰ or against legitimate queries, then so is FE.

Proof. Let Q be the number of different sets $\mathcal{U} \subseteq \mathsf{ID}$ that occur in an encryption or key generation query and let $\mathcal{U}_1, \ldots, \mathcal{U}_Q$ denote these sets in the order of their first appearance. We prove the proposition via a series of hybrids $\mathsf{G}_0^{(b)}, \ldots, \mathsf{G}_Q^{(b)}$ where $\mathsf{G}_\ell^{(b)}$, for $\ell \in [0; Q]$ and $b \in \{0, 1\}$, is the same as $\mathbf{Exp}_{\mathsf{FE}, \mathsf{fdyn}, \mathcal{A}}^{\mathsf{ddfe-}b}(1^\lambda)$ except that, upon receiving a key generation query

$$\mathcal{O}\mathsf{KeyGen}\big(i,(k_{i,\mathrm{pri}}^{(0)},(k_{i,\mathrm{pub}},\mathcal{U}_{\ell'},\mathsf{tag-f})),(k_{i,\mathrm{pri}}^{(1)},(k_{i,\mathrm{pub}},\mathcal{U}_{\ell'},\mathsf{tag-f}))\big)$$

the simulator computes and sends

$$\mathsf{DK}_i \leftarrow \begin{cases} \mathsf{mKeyGen}(\mathsf{mSK}_i, \mathsf{tag-f}, (k_{i, \mathrm{pri}}^{(0)}, k_{i, \mathrm{pub}})) & \text{if } \ell' \leq \ell \\ \mathsf{mKeyGen}(\mathsf{mSK}_i, \mathsf{tag-f}, (k_{i, \mathrm{pri}}^{(b)}, k_{i, \mathrm{pub}})) & \text{if } \ell' > \ell \end{cases},$$

¹⁰ An encryption (resp. key generation) query for a tuple (i, tag) (resp. (i, tag-f)) is said to be a *repetition* if an encryption (resp. key generation) query for the same tuple (i, tag) (resp. (i, tag-f)) has already been submitted before.

and upon receiving an encryption query

$$\mathcal{O}\mathsf{Enc}(i, (m_{i, \mathrm{pri}}^{(0)}, (m_{i, \mathrm{pub}}, \mathcal{U}_{\ell'}, \mathsf{tag})), (m_{i, \mathrm{pri}}^{(1)}, (m_{i, \mathrm{pub}}, \mathcal{U}_{\ell'}, \mathsf{tag}))) \ ,$$

the simulator computes and sends

$$\mathsf{CT}_i \leftarrow \begin{cases} \mathsf{mEnc}(\mathsf{mSK}_i, \mathsf{tag}, (m_{i, \mathrm{pri}}^{(0)}, m_{i, \mathrm{pub}})) & \text{if } \ell' \leq \ell \\ \mathsf{mEnc}(\mathsf{mSK}_i, \mathsf{tag}, (m_{i, \mathrm{pri}}^{(b)}, m_{i, \mathrm{pub}})) & \text{if } \ell' > \ell \end{cases}$$

Below, we prove the following claim for all $\ell \in [Q]$.

Claim 27. If mFE is \mathbb{A} -dynamizable and stat-yyy-zzz-secure, $\{\mathsf{PRF}_K\}_{K\in\mathcal{K}}$ and $\{\mathsf{PRF}'_{K'}\}_{K'\in\mathcal{K}'}$ are pseudorandom and NIKE is secure, then we have $\mathsf{G}^{(b)}_{\ell-1} \approx_c \mathsf{G}^{(b)}_{\ell}$.

Furthermore, we note that $\mathsf{G}_{0}^{(b)} = \mathbf{Exp}_{\mathsf{FE}, f^{\mathsf{dyn}}, \mathcal{A}}^{\mathsf{ddfe}, b}(1^{\lambda})$, for $b \in \{0, 1\}$, and $\mathsf{G}_{Q}^{(0)} \equiv \mathsf{G}_{Q}^{(1)}$ because the adversary's view is independent of the bit b. This concludes the proof of the proposition. \Box

We now prove the claim.

Proof (of Claim 27). The proof is a sequence of hybrids $\widehat{\mathsf{G}}_{0}^{(\beta)}, \ldots, \widehat{\mathsf{G}}_{3}^{(\beta)}$ for $\beta \in \{0, 1\}$.

Game $\widehat{\mathsf{G}}_{0}^{(\beta)}$ for $\beta \in \{0,1\}$: This is game $\mathsf{G}_{\ell-1+\beta}^{b}$. In particular, upon receiving an encryption query $\mathcal{O}\mathsf{Enc}(i, m_{i}^{(0)}, m_{i}^{(1)})$ with $m_{i}^{\gamma} = (m_{i, \mathrm{pri}}^{\gamma}, m_{i, \mathrm{pub}} = (m_{i, \mathrm{pub}}^{\prime}, \mathcal{U}_{\ell}, \mathsf{tag}))$ for $\gamma \in \{0, 1\}$ such that $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$, the challenger computes

$$\begin{aligned} \forall j \in \mathcal{U}_{\ell} \setminus \{i\} \colon & K'_{i,j} \leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_{i},\mathsf{nPK}_{j}) \\ s_{i} &= \sum_{j \in \mathcal{U}_{\ell} \setminus \{i\}} (-1)^{j < i} \mathsf{PRF}'_{K'_{i,j}}(\mathcal{U}_{\ell}) \\ & \mathsf{mSK}_{i} \leftarrow \mathsf{mSetupUser}(\mathsf{mPP},s_{i};\mathsf{PRF}_{K_{i}}(\mathcal{U}_{\ell})) \\ & \mathsf{CT}_{i} \leftarrow \mathsf{mEnc}(\mathsf{mSK}_{i},\mathsf{tag},(m^{(b')}_{i},m'_{i},\mathsf{pub})) \end{aligned}$$

where b' = b if $\beta = 0$ and b' = 0 if $\beta = 1$. Similarly, for a key generation query $\mathcal{O}\mathsf{KeyGen}(i, k_i^{(0)}, k_i^{(1)})$ with $k_i^{\gamma} = (k_{i, \text{pri}}^{\gamma}, k_{i, \text{pub}} = (k_{i, \text{pub}}^{\prime}, \mathcal{U}_{\ell}, \mathsf{tag-f}))$ for $\gamma \in \{0, 1\}$ such that $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$, it computes

$$\begin{split} \forall j \in \mathcal{U}_{\ell} \setminus \{i\} \colon & K'_{i,j} \leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_{i},\mathsf{nPK}_{j}) \\ & s_{i} = \sum_{j \in \mathcal{U}_{\ell} \setminus \{i\}} (-1)^{j < i} \mathsf{PRF}'_{K'_{i,j}}(\mathcal{U}_{\ell}) \\ & \mathsf{mSK}_{i} \leftarrow \mathsf{mSetupUser}(\mathsf{mPP},s_{i};\mathsf{PRF}_{K_{i}}(\mathcal{U}_{\ell})) \\ & \mathsf{DK}_{i} \leftarrow \mathsf{mKeyGen}(\mathsf{mSK}_{i},\mathsf{tag-f},(k^{(b')}_{i,\mathrm{pri}},k'_{i,\mathrm{pub}})) \end{split}$$

Game $\widehat{\mathsf{G}}_{1}^{(\beta)}$ for $\beta \in \{0, 1\}$: This game is the same as $\widehat{\mathsf{G}}_{0}^{(\beta)}$ except that the challenger initially samples $t_{i} \stackrel{\text{\tiny \$}}{\leftarrow} \mathbb{A}$ for all $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$ conditioned on $\sum_{i \in \mathcal{U}_{\ell} \cap \mathcal{H}} t_{i} = -\sum_{i \in \mathcal{U}_{\ell} \setminus \mathcal{H}} \sum_{j \in \mathcal{U}_{K} \setminus \{i\}} (-1)^{j < i} \mathsf{PRF}'_{K'_{i,j}}(\mathcal{U}_{\ell})$. Upon receiving an encryption or key generation query with respect to \mathcal{U}_{ℓ} and $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$, the challenger computes

$$\mathsf{mSK}_i \leftarrow \mathsf{mSetupUser}(\mathsf{mPP}, |t_i|; \mathsf{PRF}_{K_i}(\mathcal{U}_\ell))$$

Below, we prove the following claim.

Claim 28. If $\{\mathsf{PRF}'_{K'}\}_{K'\in\mathcal{K}'}$ is pseudorandom and NIKE is secure, then we have $\widehat{\mathsf{G}}_{0}^{(\beta)}\approx_{c}\widehat{\mathsf{G}}_{1}^{(\beta)}$.

Game $\widehat{\mathsf{G}}_{2}^{(\beta)}$ for $\beta \in \{0,1\}$: This game is the same as $\widehat{\mathsf{G}}_{1}^{(\beta)}$ except that the challenger initially samples $r_{\ell,i} \stackrel{\text{\tiny (s)}}{\leftarrow} \{0,1\}^{\text{poly}(\lambda)}$ for all $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$. Upon receiving an encryption or key generation query with respect to \mathcal{U}_{ℓ} and $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$, the challenger computes

$$\mathsf{mSK}_i \leftarrow \mathsf{mSetupUser}(\mathsf{mPP}, t_i; [r_{\ell,i}])$$

We have $\widehat{\mathsf{G}}_{1}^{(\beta)} \approx_{c} \widehat{\mathsf{G}}_{2}^{(\beta)}$ under the pseudorandomness of PRF. **Game** $\widehat{\mathsf{G}}_{3}^{(\beta)}$ for $\beta \in \{0, 1\}$: This game is the same as $\widehat{\mathsf{G}}_{2}^{(\beta)}$ except that upon receiving an encryption or key generation query with respect to \mathcal{U}_{ℓ} and $i \in \mathcal{U}_{\ell} \cap \mathcal{H}$, the challenger respectively computes

$$\begin{split} \mathsf{DK}_i &\leftarrow \mathsf{mKeyGen}(\mathsf{mSK}_i, \mathsf{tag-f}, (\fbox{k_{i,\mathrm{pri}}^{(0)}}, k_{i,\mathrm{pub}})) \\ \mathsf{CT}_i &\leftarrow \mathsf{mEnc}(\mathsf{mSK}_i, \mathsf{tag}, (\fbox{m_{i,\mathrm{pri}}^{(0)}}, m_{i,\mathrm{pub}})) \end{split}$$

We have $\widehat{\mathsf{G}}_{2}^{\scriptscriptstyle(\beta)} \approx_{c} \widehat{\mathsf{G}}_{3}^{\scriptscriptstyle(\beta)}$ under the dynamizability and security of mFE. Moreover, we can observe that $\widehat{\mathsf{G}}_{3}^{(0)} \approx_{c} \widehat{\mathsf{G}}_{3}^{(1)}$, as the adversaries view is independent of the bit β .

Proof (of Claim 28). Let $\mathcal{U}_{\ell} \cap \mathcal{H} = \{i_1, \ldots, i_N\}$. We consider a series of hybrids $\overline{\mathsf{G}}_0, \ldots, \overline{\mathsf{G}}_{N-1}$. For each $\kappa \in [0; N-1], \overline{\mathsf{G}}_{\kappa}$ is the same $\widehat{\mathsf{G}}_0^{(\beta)}$ except that the challenger initially samples $s_{i_{\nu}, i_N} \stackrel{\text{\tiny \ensuremath{\&}}}{=} \mathbb{A}$ for all $\nu \in [\kappa]$ and, upon receiving an encryption or key generation query with respect to \mathcal{U}_{ℓ} and i_{ν} for $\nu \in [N]$ (note that replies to other oracle queries do not change between $\widehat{\mathsf{G}}_{0}^{(\beta)}$ and $\widehat{\mathsf{G}}_{1}^{(\beta)}$), the challenger computes

$$s_{i\nu} = \begin{cases} \sum_{j \in \mathcal{U}_K \setminus \{i_\nu, i_N\}} (-1)^{j < i_\nu} \mathsf{PRF}'_{K'_{i\nu,j}}(\mathcal{U}_\ell) + s_{i_\nu, i_N} & \text{if } \nu \le \kappa \\ \sum_{j \in \mathcal{U}_K \setminus \{i_\nu\}} (-1)^{j < i_\nu} \mathsf{PRF}'_{K'_{i\nu,j}}(\mathcal{U}_\ell) & \text{if } \kappa < \nu < N \\ \sum_{j \in \mathcal{U}_K \setminus \{i_\eta\}_{\eta \in [\kappa] \cup \{N\}}} (-1)^{j < i_N} \mathsf{PRF}'_{K'_{i_N,j}}(\mathcal{U}_\ell) - \sum_{\eta \in [\kappa]} s_{i_\eta, i_N} & \text{if } \nu = N \end{cases}$$

We can observe that $\overline{\mathsf{G}}_0 = \widehat{\mathsf{G}}_0^{(\beta)}$ and $\overline{\mathsf{G}}_{N-1} \equiv \widehat{\mathsf{G}}_1^{(\beta)}$. Thus, the remaining task is to prove $\overline{\mathsf{G}}_{\kappa-1} \approx_c \overline{\mathsf{G}}_{\kappa}$ for $\kappa \in [N-1]$. To do so, we first change the way of choosing K'_{i_{κ},i_N} to $K'_{i_{\kappa},i_N} \stackrel{\hspace{0.1em}\hspace{0.1em}}{\leftarrow} \mathcal{K}'$ instead of $K'_{i_{\kappa},N} \leftarrow \mathsf{nSharedKey}(\mathsf{nSK}_{i_{\kappa}},\mathsf{nPK}_{i_{N}})$. This cannot be noticed by the adversary under the INDsecurity of NIKE. Subsequently, we conclude $\overline{\mathsf{G}}_{\kappa-1} \approx_c \overline{\mathsf{G}}_{\kappa}$ from the security of PRF'.

Concrete Instantiations. All schemes we instantiate satisfy the dynamizability property. Particularly, this includes our DMCFE for AB-AWS presented in Section A.2, several existing schemes in the literature [CDG⁺18a, LT19, NPS24] for the inner product functionality (with standard or function-hiding security). The latter [CDG⁺18a, LT19, NPS24] come with a few restrictions in their security model and as in [NPS24] we remove all of them except for the constraints on repetitions, using a generic conversion (see Appendix A.1). These new schemes have interesting, previously unattained properties which we highlight by a frame

Theorem 29. • Conversion of our DMCFE for AB-AWS. Assuming SXDH in the ROM, there exists a DDFE scheme for $|f^{ab-aws}|$ that is stat-sel-sym-secure against legitimate queries. For details, see Section A.2. In Section $\overline{5}$, we show how to remove the restriction to legitimate queries under certain conditions.

- Conversion of the DMCFE of [CDG⁺18a]. Assuming SXDH in the ROM, there exists a DDFE scheme for f^{ip} that is stat-adap-sym-secure without repetitions. For details, see Section A.3.
- Conversion of the DMCFE of [LT19]. Assuming LWE in the standard model, there exists a DDFE scheme for f^{ip} that is stat-adap-sym-secure without repetitions. For details, see Section A.4.
- Conversion of the DMCFE of [NPS24]. Assuming SXDH in the ROM, there exists a DDFE scheme for f^{fh-ip} that is stat-adap-sym-secure with bounded repetitions for OKeyGen queries and unbounded repetitions for OEnc queries. For details, see Section A.5.

5 Security Against Any Queries

In this section, we show how the restriction to legitimate queries can be removed under certain conditions. For notational convenience, we present our construction in the particular case of DDFE for AB-AWS. Nonetheless, the conversion may also be applied to other attribute-based functionalities "AB-XXX", where XXX can be more general than AWS.

5.1 Attribute-Based All-or-Nothing Encapsulation

We define a new functionality for DDFE which is a generalization of the All-or-Nothing Encapsulation (AoNE) introduced in [CDSG⁺20].

Definition 30 (Attribute-Based AoNE (AB-AoNE)). For $\lambda \in \mathbb{N}$, let $\mathsf{Tag}_{\lambda} = \mathsf{ID}_{\lambda} = \mathcal{R}_{\lambda} = \{0,1\}^{\mathrm{poly}(\lambda)}$, $\mathcal{K}_{\lambda,\mathrm{pub}} = \mathcal{F}_{n'_{0},1}^{\mathsf{abp}} \times 2^{\mathsf{ID}_{\lambda}}$, $\mathcal{K}_{\lambda,\mathrm{pri}} = \{\top\}$, $\mathcal{M}_{\lambda,\mathrm{pub}} = (\mathbb{Z}_{q_{\lambda}}^{n'_{0}} \cup \{\star\}) \times 2^{\mathsf{ID}_{\lambda}} \times \mathsf{Tag}_{\lambda}$ and $\mathcal{M}_{\lambda,\mathrm{pri}} = \{0,1\}^{L}$ for polynomials $n'_{0} = n'_{0}(\lambda)$, $L = L(\lambda) \colon \mathbb{N} \to \mathbb{N}$. The AB-AoNE functionality $f^{\mathsf{ab-aone}} = \{f_{\lambda}^{\mathsf{ab-aone}} \colon \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_{\lambda} \times \mathcal{K}_{\lambda})^{n} \times \bigcup_{n \in \mathbb{N}} (\mathsf{ID}_{\lambda} \times \mathcal{M}_{\lambda})^{n} \to \mathcal{R}_{\lambda}\}_{\lambda \in \mathbb{N}}$ is defined via

$$f_{\lambda}^{\text{ab-aone}}((i,k_i)_{i \in \mathcal{U}_K},(i,m_i)_{i \in \mathcal{U}_M}) = \begin{cases} (x_i)_{i \in \mathcal{U}} & \text{if condition (*) holds} \\ \bot & \text{otherwise} \end{cases}$$

for all $\lambda \in \mathbb{N}$, where condition (*) holds if $\mathcal{U}_M = \mathcal{U}_K$ (in which case we define $\mathcal{U} \eqqcolon \mathcal{U}_K$) and there exists $\mathsf{tag} \in \mathsf{Tag}_\lambda$ such that for each $i \in \mathcal{U}$: (i) m_i is of the form $(m_{i,\mathrm{pri}} \coloneqq x_i, m_{i,\mathrm{pub}} \coloneqq (\mathbf{y}_i, \mathcal{U}, \mathsf{tag}))$, (ii) k_i is of the form $(k_{i,\mathrm{pri}} \coloneqq \top, k_{i,\mathrm{pub}} \coloneqq (g_i, \mathcal{U}))$, and (iii) $g_i(\mathbf{y}_i) = 0$.

Construction 31. The construction uses the following ingredients:

- A lockable obfuscation scheme LObf = (Obf, Eval) with lock space $\mathcal{L} = \{0, 1\}^L$ for some polynomial $L = L(\lambda)$.
- An ABE scheme aFE = (aSetup, aKeyGen, aEnc, aDec) for the policy class $\mathcal{F}_{n'_0,1}^{abp}$ with message space \mathcal{L} .
- An IBE scheme idFE = (idSetup, idKeyGen, idEnc, idDec) with identity space $2^{ID} \times Tag$.
- A PRF family $\{\mathsf{PRF}_K\}_{K \in \mathcal{K}}$ with key space \mathcal{K} .

We implicitly assume that each $\mathcal{U} \subseteq \mathsf{ID}$ is equipped with a universally accepted cyclic¹¹ permutation $\pi_{\mathcal{U}}: \mathcal{U} \to \mathcal{U}$. If \mathcal{U} is clear from the context (e.g. in the description of an algorithm that takes a set $\mathcal{U} \subseteq \mathsf{ID}$ as input), we use the following shorthand notation: given $i \in \mathsf{ID}$ and $k \in \mathbb{N}$, we write $i \triangleright k \coloneqq \pi_{\mathcal{U}}^k(i)$ for the result of k successive applications of $\pi_{\mathcal{U}}$ on input i, i.e. $i \triangleright k$ is the k-th successor of i under $\pi_{\mathcal{U}}$. In particular, we observe that $i \triangleright 0 = i \triangleright |\mathcal{U}| = i \triangleright 2 \cdot |\mathcal{U}| = i$.

The details of the DDFE scheme anFE for AB-AoNE go as follows:

¹¹ We call a permutation cyclic if it consists of a single cycle *without* fixed points.

GSetup (1^{λ}) : On input the security parameter 1^{λ} , output $\mathsf{PP} \coloneqq 1^{\lambda}$.

- LSetup(PP, *i*): On input the public parameters PP and a user $i \in ID$, sample $K_i \stackrel{\text{\tiny \sc s}}{\leftarrow} \mathcal{K}$, generate (idMPK_i, idMSK_i) \leftarrow idSetup(1^{λ}) and output the key pair (SK_i := (idMSK_i, K_i), pk_i := idMPK_i).
- KeyGen(SK_i, k_i): On input a secret key SK_i = (idMSK_i, K_i) and $k_i = (k_{i,pri} = \top, k_{i,pub} = (g_i, \mathcal{U}_K))$, compute and output DK_i := aDK_i as follows:

 $(\mathsf{aMPK}_i, \mathsf{aMSK}_i) \leftarrow \mathsf{aSetup}(1^{\lambda}; \mathsf{PRF}_{K_i}(\mathcal{U}_K))$, $\mathsf{aDK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, g_i)$

Enc(SK_i, m_i): On input a secret key SK_i = (idMSK_i, K_i) and $m_i = (m_{i,pri}, m_{i,pub} = (\mathbf{y}_i, \mathcal{U}_M, tag))$ such that $i \in \mathcal{U}_M$, sample $\sigma_i \stackrel{*}{\leftarrow} \mathcal{L}$ and output CT_i := \widetilde{C}_i generated as follows:

$$\begin{aligned} (\mathsf{aMPK}_{i}, \mathsf{aMSK}_{i}) &\leftarrow \mathsf{aSetup}(1^{\lambda}; \mathsf{PRF}_{K_{i}}(\mathcal{U}_{M})) \\ c_{i}^{(0)} &\leftarrow \mathsf{aEnc}(\mathsf{aMPK}_{i}, \mathbf{y}_{i}, \sigma_{i}) \\ \forall k = 1, \dots, |\mathcal{U}_{M}| - 1; \quad c_{i}^{(k)} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_{i \triangleright k}, (\mathcal{U}_{M}, \mathsf{tag}), c_{i}^{(k-1)}) \\ \mathsf{idDK}_{i} \leftarrow \mathsf{idKeyGen}(\mathsf{idMSK}_{i}, (\mathcal{U}_{M}, \mathsf{tag})) \\ \widetilde{C}_{i} \leftarrow \mathsf{Obf}(1^{\lambda}, C_{\mathcal{U}_{M}, i}[\mathsf{idDK}_{i}, c_{i}^{(n-1)}], (\mathsf{idDK}_{i}, m_{i, \mathrm{pri}}), \sigma_{i}) \end{aligned}$$

 $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in\mathcal{U}_K},\{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}): \text{ On input a set of decryption keys } \{\mathsf{DK}_i\}_{i\in\mathcal{U}_K} \text{ and a set of ciphertexts } \{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}, \text{ if } \mathcal{U}_K \neq \mathcal{U}_M \text{ abort with failure. Otherwise, parse } \mathsf{DK}_i = \mathsf{a}\mathsf{DK}_i \text{ and } \mathsf{CT}_i = \widetilde{C}_i \text{ for all } i\in\mathcal{U} := \mathcal{U}_M, \text{ then output } \{m_{i,\mathrm{pri}}\}_{i\in\mathcal{U}} \text{ computed as follows:} \end{cases}$

$$\forall i \in \mathcal{U}: \quad (\mathsf{idDK}_i, m_{i, \mathrm{pri}}) \leftarrow \mathsf{Eval}(\hat{C}_i, (n, \{\hat{C}_j\}_{j \in \mathcal{U} \setminus \{i\}}, \emptyset, \{\mathsf{aDK}_j\}_{j \in \mathcal{U}}))$$

input : • a number $t \in [n]$ where $n \coloneqq |\mathcal{U}|$ • a set of obfuscated circuits $\{\tilde{C}_{i \triangleright \ell}\}_{\ell \in [t-1]}$ • a set of idFE decryption keys $\{\mathsf{idDK}_{i \triangleright \ell}\}_{\ell \in [t;n-1]}$ • a set of aFE decryption keys $\{aDK_{i \geq \ell}\}_{\ell \in [0,t-1]}$ **output**: an element of the lock space $\sigma_i \in \mathcal{L}$ or \perp initialize {idDK_{$i \triangleright \ell$} $\leftarrow \bot$ }_{$\ell \in [t-1]}</sub>$ for $k \leftarrow 1$ to t - 1 do $d_k \leftarrow \mathsf{Eval}(\widetilde{C}_{i \triangleright k}, (t-k, \{\widetilde{C}_{i \triangleright \ell}\}_{\ell \in [k+1;t-1]}, \{\mathsf{id}\mathsf{DK}_{i \triangleright \ell}\}_{\ell \in [t;n+k-1]}, \{\mathsf{a}\mathsf{DK}_{i \triangleright \ell}\}_{\ell \in [k;t-1]}))$ if $d_k = \perp$ then return \perp else parse (idDK_{$i \triangleright k$}, $m_{i \triangleright k, \text{pri}}$) $\leftarrow d_k$ end for $k \leftarrow n-1$ to 1 do $| c_i^{(k-1)} \leftarrow \mathsf{idDec}(\mathsf{idDK}_{i \triangleright k}, c_i^{(k)})$ end return $\sigma_i \leftarrow \mathsf{aDec}(\mathsf{aDK}_i, c_i^{(0)})$

Fig. 5: Definition of the circuit $C_{\mathcal{U},i}$ with hardwired values $[\mathsf{idDK}_i, c_i^{(n-1)}]$ and input $(t, \{\widetilde{C}_{i \triangleright \ell}\}_{\ell \in [t-1]}, \{\mathsf{idDK}_{i \triangleright \ell}\}_{\ell \in [t;n-1]}, \{\mathsf{adK}_{i \triangleright \ell}\}_{\ell \in [0;t-1]})$

Correctness and Security. Let $\lambda \in \mathbb{N}$, $\mathcal{U} \subseteq ID$, tag \in Tag and, for each $i \in \mathcal{U}$, $m_{i,pri} \in \mathcal{M}_{pri}$, $\mathbf{y}_i \in \mathbb{Z}_{q_\lambda}^{n'_0} \cup \{\star\}$ and $f_i \in \mathcal{F}_{n'_0,1}^{\mathsf{abp}}$ such that $f_i(\mathbf{y}_i) = 0$. Furthermore, let $\{\mathsf{DK}_i = \mathsf{aDK}_i\}_{i \in \mathcal{U}}$ and $\{\mathsf{CT}_i = \widetilde{C}_i\}_{i \in \mathcal{U}}$ be created as in Construction 31. To establish correctness, we show the following statement.

Proposition 32. For $i \in U$, $n \coloneqq |U|$, $t \in [n]$ and $k \in [0; t-1]$, we have

$$\begin{aligned} \mathsf{Eval}(\widetilde{C}_{i \triangleright k}, (t-k, \{\widetilde{C}_{i \triangleright \ell}\}_{\ell \in [k+1;t-1]}, \{\mathsf{idDK}_{i \triangleright \ell}\}_{\ell \in [t;n+k-1]}, \{\mathsf{aDK}_{i \triangleright \ell}\}_{\ell \in [k;t-1]})) \\ &= (\mathsf{idDK}_{i \triangleright k}, m_{i \triangleright k, \mathrm{pri}}) \end{aligned}$$

In particular, correctness follows from t = n and k = 0:

$$\mathsf{Eval}(\tilde{C}_i, (n, \underbrace{\{\tilde{C}_{i \triangleright \ell}\}_{\ell \in [n-1]}}_{=\{\tilde{C}_j\}_{j \in \mathcal{U} \setminus \{i\}}}, \varnothing, \underbrace{\{\mathsf{aDK}_{i \triangleright \ell}\}_{\ell \in [0; n-1]}}_{=\{\mathsf{aDK}_j\}_{j \in \mathcal{U}}})) = (\mathsf{idDK}_i, m_{i, \mathrm{pri}})$$

Due to the nested evaluations, decryption runs in time $O(n^n)$ for $n = |\mathcal{U}|$. So the scheme is efficient for constant-size sets of users, *i.e.* n = O(1).

Proof. We prove the lemma by induction over $\delta := t - k$ from 1 to n.

Base Case For $\delta = 1$, we must argue that

$$\mathsf{Eval}(C_{i \triangleright k}, (1, \varnothing, \{\mathsf{idDK}_{i \triangleright \ell}\}_{\ell \in [k+1; n+k-1]}, \{\mathsf{aDK}_{i \triangleright k}\})) = (\mathsf{idDK}_{i \triangleright k}, m_{i \triangleright k, \mathrm{pri}}) .$$

By construction, $\widetilde{C}_{i \triangleright k}$ is generated as follows:

$$\widetilde{C}_{i \triangleright k} \leftarrow \mathsf{Obf}(1^{\lambda}, C_{\mathcal{U}_{M}, i \triangleright k}[\mathsf{idDK}_{i \triangleright k}, c_{i \triangleright k}^{(n-1)}], (\mathsf{idDK}_{i \triangleright k}, m_{i \triangleright k, \mathrm{pri}}), \sigma_{i \triangleright k})$$

where $\sigma_{i \triangleright k} \stackrel{\text{s}}{\leftarrow} \mathcal{L}$ is the lock value of the obfuscated circuit. On input $(1, \emptyset, \{idDK_{i \triangleright \ell}\}_{\ell \in [k+1;n+k-1]}, \{aDK_{i \triangleright k}\})$, the circuit $C_{\mathcal{U}_M, i \triangleright k}[idDK_{i \triangleright k}, c_{i \triangleright k}^{(n-1)}]$ skips the first for loop and directly starts to remove the (n-1) layers of identity-based encryption from $c_{i \triangleright k}^{(n-1)}$ using the keys $\{idDK_{i \triangleright \ell}\}_{\ell \in [k+1;n+k-1]}$ given as input. Then it decrypts $c_{i \triangleright k}^{(0)}$ using the decryption key $aDK_{i \triangleright k}$ which is also given as input. By assumption, we have $f_{i \triangleright k}(\mathbf{y}_{i \triangleright k}) = 0$, so decryption succeeds and returns $\sigma_{i \triangleright k}$, which is also the output of the circuit. As a consequence, the obfuscated circuit is unlocked and the message $(idDK_{i \triangleright k}, m_{i \triangleright k, pri})$ is released.

Induction Step Let $\delta \in [2; n]$. We show that, if

$$\begin{aligned} \mathsf{Eval}\Big(\widetilde{C}_{i\triangleright k}, \big(\delta', \{\widetilde{C}_{i\triangleright \ell}\}_{\ell\in[k+1;k+\delta'-1]}, \{\mathsf{id}\mathsf{D}\mathsf{K}_{i\triangleright \ell}\}_{\ell\in[k+\delta';n+k-1]}, \\ \{\mathsf{a}\mathsf{D}\mathsf{K}_{i\triangleright \ell}\}_{\ell\in[k;k+\delta'-1]}\Big)\Big) &= (\mathsf{id}\mathsf{D}\mathsf{K}_{i\triangleright k}, m_{i\triangleright k, \mathrm{pri}})\end{aligned}$$

for all $\delta' \in [\delta - 1]$, then

$$\begin{aligned} \mathsf{Eval}(\tilde{C}_{i \triangleright k}, (\delta, \{\tilde{C}_{i \triangleright \ell}\}_{\ell \in [k+1;k+\delta-1]}, \{\mathsf{idDK}_{i \triangleright \ell}\}_{\ell \in [k+\delta;n+k-1]}, \\ \{\mathsf{aDK}_{i \triangleright \ell}\}_{\ell \in [k;k+\delta-1]})) &= (\mathsf{idDK}_{i \triangleright k}, m_{i \triangleright k, \mathrm{pri}}) \end{aligned}$$

By construction, $\widetilde{C}_{i\triangleright k}$ is an obfuscation of the circuit $C_{\mathcal{U}_M,i\triangleright k}[\mathsf{idDK}_{i\triangleright k}, c_{i\triangleright k}^{(n-1)}]$ under the lock value $\sigma_{i\triangleright k}$. On input $(\delta, \{\widetilde{C}_{i\triangleright \ell}\}_{\ell \in [k+1;k+\delta-1]}, \{\mathsf{idDK}_{i\triangleright \ell}\}_{\ell \in [k+\delta;n+k-1]}, \{\mathsf{aDK}_{i\triangleright \ell}\}_{\ell \in [k;k+\delta-1]})$, the circuit starts by evaluating the obfuscated circuits $\{\widetilde{C}_{i\triangleright \ell}\}_{\ell \in [k+1;k+\delta-1]}$. Note that all these evaluations proceed

with respect to inputs that are covered by the induction hypothesis. Therefore, we can conclude that after finishing the first for loop, all decryption keys $\{idDK_{i>\ell}\}_{\ell\in[k+1;n+k-1]}$ are known. (Either they were given as input or recovered during the nested evaluations). Then we can argue as in the base case: while iterating over the second for loop, the (n-1) layers of identity-based encryption are removed. Finally, $c_{i>k}^{(0)}$ can be decrypted using $aDK_{i>k}$, which is possible as $f_{i>k}(\mathbf{y}_{i>k}) = 0$ by assumption. Therefore, $\tilde{C}_{i>k}$ is unlocked and the message $(idDK_{i>k}, m_{i>k,pri})$ is released. \Box

The construction is semi-adaptively secure under static corruptions.

Proposition 33. If LObf, aFE and idFE are secure, then the DDFE scheme FE in Construction 31 is stat-adap-sym-secure.

Proof. Let q denote the number of distinct tuples $(\mathcal{U}_M, \mathsf{tag}) \in 2^{\mathsf{ID}} \times \mathsf{Tag}$ that occur as part of the public input $m_{i,\mathsf{pub}} = (\bigstar, \mathcal{U}_M, \mathsf{tag})^{12}$ of an encryption query $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$. We denote them by $(\mathcal{U}_1, \mathsf{tag}_1), \ldots, (\mathcal{U}_q, \mathsf{tag}_q)$ in the order of their first appearance. We consider a series of hybrids $\mathsf{G}_0^{(b)}, \ldots, \mathsf{G}_q^{(b)}$ where, for $\ell \in [0; q]$ and $b \in \{0, 1\}, \mathsf{G}_\ell^{(b)}$ is the same as the experiment $\mathsf{Exp}_{\mathsf{FE}, f^{\mathsf{ab-aone}}, \mathcal{A}}^{\mathsf{ddfe}, b}$ except that the challenger returns $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{SK}_i, m_i^{(0)})$ for all encryption queries $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ where the public input is of the form $m_{i,\mathsf{pub}} = (\bigstar, \mathcal{U}_j, \mathsf{tag}_j)$ such that $j \in [\ell]$.

Below we prove the following claim for all $\ell \in [q]$ and $b \in \{0, 1\}$.

Claim 34. If aFE and LObf are secure, then $\mathsf{G}_{\ell-1}^{(b)} \approx_c \mathsf{G}_{\ell}^{(b)}$.

Furthermore, we can observe that $G_0^b = \operatorname{Exp}_{\mathsf{FE}, f^{\mathsf{ab-aone}}, \mathcal{A}}^{\mathsf{ddfe}-b}(1^\lambda)$, for $b \in \{0, 1\}$, and $G_q^{(0)} \equiv G_q^{(1)}$ as the replies of the challenger in G_q^b are independent of the challenge bit $b \in \{0, 1\}$. This concludes the proof of the proposition.

We now turn to the claim.

Proof (of Claim 34). We define an event E_{ℓ} as follows:

$$\begin{split} E_{\ell}: \mbox{ For all } i \in \mathcal{U}_{\ell} \setminus \mathcal{C}, \mbox{ there exist } (i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\text{key}} \mbox{ with } k_{i, \text{pub}} = (f_i, \mathcal{U}_{\ell}) \mbox{ and } (i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\text{enc}} \mbox{ with } m_{i, \text{pub}} = (\mathbf{y}_i, \mathcal{U}_{\ell}, \text{tag}_{\ell}) \mbox{ such that } f_i(\mathbf{y}_i) = 0. \end{split}$$

Intuitively, E_{ℓ} occurs if queries with respect to $(\mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$ are legitimate. The proof is done via a sequence of hybrids $\widehat{\mathsf{G}}_{0}^{\beta}, \ldots, \widehat{\mathsf{G}}_{3}^{\beta}$ for $\beta \in \{0, 1\}$.

Game $\widehat{\mathsf{G}}_{0}^{(\beta)}$ for $\beta \in \{0,1\}$ This is game $\mathsf{G}_{\ell-1+\beta}^{(b)}$.

- **Game** $\widehat{\mathsf{G}}_{1}^{(\beta)}$ for $\beta \in \{0,1\}$ This is the same as $\mathsf{G}_{0}^{(\beta)}$ except that the challenger chooses a random bit $d \stackrel{\hspace{0.1em}{\leftarrow}}{\leftarrow} \{0,1\}$ during Initialize. Upon \mathcal{A} calling Finalize, if $[d=0 \text{ and } E_{\ell} \text{ occurs}]$ or $[d=1 \text{ and } E_{\ell} \text{ does not occur}]$, the simulator outputs 0. It is not hard to see that $\widehat{\mathsf{G}}_{0}^{(\beta)} \approx_{s} \widehat{\mathsf{G}}_{1}^{(\beta)}$. We emphasize the fact that the guess in $\widehat{\mathsf{G}}_{1}^{(\beta)}$ doubles the adversary's advantage in the transitions $\widehat{\mathsf{G}}_{1}^{(\beta)} \to \widehat{\mathsf{G}}_{2}^{(\beta)}$ and $\widehat{\mathsf{G}}_{2}^{(\beta)} \to \widehat{\mathsf{G}}_{3}^{(\beta)}$. However, the sequence of games is designed in such a way that the simulator never has to guess more than once, so globally the advantage grows only linearly, not exponentially in $q = \operatorname{poly}(\lambda)$.
- **Game** $\widehat{\mathsf{G}}_{2}^{(\beta)}$ for $\beta \in \{0,1\}$ This game is the same as $\widehat{\mathsf{G}}_{1}^{(\beta)}$ except that if d = 1, then the challenger returns $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{SK}_i, m_i^{(0)})$ for all encryption queries $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ such that $m_{i, \text{pub}} = (\overleftrightarrow{\approx}, \mathcal{U}_\ell, \mathsf{tag}_\ell)$. Below, we prove the following claim.

Claim 35. If aFE and LObf are secure, then $\widehat{\mathsf{G}}_{1}^{(\beta)} \approx_{c} \widehat{\mathsf{G}}_{2}^{(\beta)}$.

¹² The symbol \Leftrightarrow is a placeholder for an arbitrary attribute \mathbf{y}_i , not the wildcard \star .

Game $\widehat{\mathsf{G}}_{3}^{(\beta)}$ for $\beta \in \{0, 1\}$ This game is the same as $\mathsf{G}_{\ell, 2}^{(b)}$ except that if d = 0, then the challenger returns $\mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{SK}_i, m_i^{(0)})$ for all encryption queries $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ such that $m_{i, \text{pub}} = (\mathbf{y}_i, \mathcal{U}_\ell, \mathsf{tag}_\ell)$. Below, we prove the following claim.

Claim 36. If aFE, idFE and LObf are secure, then $\widehat{\mathsf{G}}_2^{\scriptscriptstyle(\beta)} \approx_c \widehat{\mathsf{G}}_3^{\scriptscriptstyle(\beta)}$.

Furthermore, we can observe that $\widehat{\mathsf{G}}_3^{(0)} \equiv \widehat{\mathsf{G}}_3^{(1)}$ which concludes the proof of the claim. \Box

Proof (of Claim 35). Note that the outcome of $\widehat{\mathsf{G}}_{1}^{(\beta)}$ and $\widehat{\mathsf{G}}_{2}^{(\beta)}$ are the same if d = 0 or E_{ℓ} does not occur. We therefore assume that d = 1 and E_{ℓ} occurs in the remainder of the proof. We start with a detailed analysis of the adversary's admissibility condition. Let $i_0 \in \mathcal{U}_{\ell} \setminus \mathcal{C}$ and consider an encryption query $\mathsf{CT}_{i_0} \leftarrow \mathcal{O}\mathsf{Enc}(i_0, m_{i_0}^{(0)}, m_{i_0}^{(1)})$ with $m_{i_0}^{(b)} = (m_{i_0, \mathrm{pri}}^{(b)}, m_{i_0, \mathrm{pub}})$, for $b \in \{0, 1\}$, and $m_{i_0, \mathrm{pub}} = (\mathbf{y}_{i_0}, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$. We observe that if there exists $(i_0, k_{i_0}^{(0)}, k_{i_0}^{(1)}) \in \mathcal{Q}_{\mathsf{key}}$ with $k_{i_0, \mathrm{pub}} = (f_{i_0}, \mathcal{U}_{\ell})$ such that $f_{i_0}(\mathbf{y}_{i_0}) = 0$, then $m_{i_0, \mathrm{pri}}^{(0)} = m_{i_0, \mathrm{pri}}^{(1)}$. Indeed, in this case the adversary can pick f_i and \mathbf{y}_i such that $f_i(\mathbf{y}_i) = 0$ for each $i \in \mathcal{U}_{\ell} \cap \mathcal{C}$, then generate

$$CT_i \leftarrow Enc(SK_i, (m_{i,pri} = \top, m_{i,pub} = (\mathbf{y}_i, \mathcal{U}_\ell, \mathsf{tag}_\ell)))$$
$$DK_i \leftarrow KeyGen(SK_i, (k_{i,pri} = \top, k_{i,pub} = (f_i, \mathcal{U}_\ell)))$$

by herself. Furthermore, the event E_{ℓ} guarantees that the adversary possesses ciphertexts CT_i for attributes \mathbf{y}_i and decryption keys DK_i for policies f_i such that $f_i(\mathbf{y}_i) = 0$ for all $i \in \mathcal{U}_{\ell} \setminus \mathcal{C}$. These ciphertexts and keys (obtained via oracle queries for $i \in \mathcal{U}_{\ell} \setminus \mathcal{C}$ or self-generated for $i \in \mathcal{U}_{\ell} \cap \mathcal{C}$) can be used for a joint decryption with the ciphertext CT_{i_0} . Then the admissibility gives $m_{i_0, \mathrm{pri}}^{(0)} = m_{i_0, \mathrm{pri}}^{(1)}$. (For $i_0 \in \mathcal{U}_{\ell} \cap \mathcal{C}$, we always have $m_{i_0, \mathrm{pri}}^{(0)} = m_{i_0, \mathrm{pri}}^{(1)}$, as we consider sym-security). The appear of the following of backwide

The proof consists of the following sequence of hybrids.

 $\begin{array}{l} \textbf{Game} \ \overline{\mathsf{G}}_{0}^{(\gamma)} \ \textbf{for} \ \gamma \in \{0,1\} \textbf{:} \ \text{This is game} \ \widehat{\mathsf{G}}_{1+\gamma}^{(\beta)} \textbf{.} \\ \textbf{Game} \ \overline{\mathsf{G}}_{1}^{(\gamma)} \ \textbf{for} \ \gamma \in \{0,1\} \textbf{:} \ \text{This is the same as } \overline{\mathsf{G}}_{0}^{(\gamma)} \ \text{except that, for each query } \mathcal{O}\mathsf{Enc}(i,m_{i}^{(0)},m_{i}^{(1)}) \\ \text{with} \ m_{i,\mathrm{pri}}^{(0)} \neq m_{i,\mathrm{pri}}^{(1)} \ \text{and} \ m_{i,\mathrm{pub}} = (\mathbf{y}_{i},\mathcal{U}_{\ell}, \mathsf{tag}_{\ell}), \ \text{the challenger computes} \end{array}$

$$c_i^{(0)} \gets \mathsf{aEnc}\big(\mathsf{aMPK}_i, \mathbf{y}_i, \boxed{\mathbf{0}^{L(\lambda)}}\big)$$

instead of $c_i^{(0)} \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_i, \mathbf{y}_i, \sigma_i)$. From the above analysis of the admissibility condition, we have $f_i(\mathbf{y}_i) = 1$ for every key generation query $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{key}$ with $k_{i,pub} = (f_i, \mathcal{U}_\ell)$. Therefore, the security of aFE implies $\overline{\mathsf{G}}_0^{(\gamma)} \approx_c \overline{\mathsf{G}}_1^{(\gamma)}$.

Game $\overline{\mathsf{G}}_{2}^{(\gamma)}$ for $\gamma \in \{0, 1\}$: This is the same as $\overline{\mathsf{G}}_{1}^{(\gamma)}$ except that, for each query $\mathcal{O}\mathsf{Enc}(i, m_{i}^{(0)}, m_{i}^{(1)})$ with $m_{i,\mathrm{pri}}^{(0)} \neq m_{i,\mathrm{pri}}^{(1)}$ and $m_{i,\mathrm{pub}} = (\mathbf{y}_{i}, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$, the challenger computes

$$\widetilde{C}_i \leftarrow \boxed{\mathsf{Sim}\big(1^{\lambda}, 1^{|C_{\mathcal{U}_M, i}[\mathsf{idDK}_i, c_i^{(n-1)}]|}, 1^{|(\mathsf{idDK}_i, m_{i, \mathrm{pri}}^{(0)})|}\big)}$$

instead of $\widetilde{C}_i \leftarrow \mathsf{Obf}(1^{\lambda}, C_{\mathcal{U}_M, i}[\mathsf{idDK}_i, c_i^{(n-1)}], (\mathsf{idDK}_i, m_{i, \mathrm{pri}}^{(b')}), \sigma_i)$, where Sim is the simulation algorithm whose existence is guaranteed by the security of LObf, and b' = b if $\beta = \gamma = 0$ and b' = 0 otherwise. It follows $\overline{\mathsf{G}}_1^{(\gamma)} \approx_c \overline{\mathsf{G}}_2^{(\gamma)}$ from the security of LObf. Furthermore, we observe that all replies to oracle queries in $\overline{\mathsf{G}}_2^{(\gamma)}$ are independent of $\gamma \in \{0, 1\}$. Therefore, we have $\overline{\mathsf{G}}_2^{(0)} \equiv \overline{\mathsf{G}}_2^{(1)}$ which concludes the proof of the claim. \Box

Proof (of Claim 36). Note that the outcome of $\widehat{\mathsf{G}}_{2}^{(\beta)}$ and $\widehat{\mathsf{G}}_{3}^{(\beta)}$ are the same if d = 1 or E_{ℓ} occurs. We therefore assume that d = 0 and E_{ℓ} does not occur in the remainder of the proof. We observe

that the negation of E_{ℓ} implies the existence of an $i_0 \in \mathcal{U}_{\ell} \setminus \mathcal{C}$ which satisfies $f_{i_0}(\mathbf{y}_{i_0}) = 1$ for each combination of $(i_0, m_{i_0}^{(0)}, m_{i_0}^{(1)}) \in \mathcal{Q}_{enc}$ and $(i_0, k_{i_0}^{(0)}, k_{i_0}^{(1)}) \in \mathcal{Q}_{key}$ with $m_{i_0, \text{pub}} = (a_{i_0}, \mathcal{U}_{\ell}, \text{tag}_{\ell})$ and $k_{i_0, \text{pub}} = (f_{i_0}, \mathcal{U}_{\ell})$.

The proof consists of the following sequence of hybrids.

Game $\overline{\mathsf{G}}_{0}^{(\gamma)}$ for $\gamma \in \{0, 1\}$: This is game $\widehat{\mathsf{G}}_{2+\gamma}^{(\beta)}$. Game $\overline{\mathsf{G}}_{1}^{(\gamma)}$ for $\gamma \in \{0, 1\}$: This is the same as $\overline{\mathsf{G}}_{0}^{(\gamma)}$ except that, for each query $\mathcal{O}\mathsf{Enc}(i_{0}, m_{i_{0}}^{(0)}, m_{i_{0}}^{(1)})$ with $m_{i_{0}, \mathrm{pub}} = (\mathbf{y}_{i_{0}}, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$, the challenger computes

$$c_{i_0}^{(0)} \leftarrow \mathsf{aEnc}\big(\mathsf{aMPK}_{i_0}, \mathbf{y}_{i_0}, \boxed{\mathbf{0}^{L(\lambda)}}\big)$$

instead of $c_{i_0}^{(0)} \leftarrow \mathsf{aEnc}(\mathsf{aMPK}_{i_0}, \mathbf{y}_{i_0}, \sigma_{i_0})$. As we have $f_{i_0}(\mathbf{y}_{i_0}) = 1$ for every key generation query $(i_0, k_{i_0}^{(0)}, k_{i_0}^{(1)}) \in \mathcal{Q}_{\mathsf{key}}$ with $k_{i_0, \mathsf{pub}} = (f_{i_0}, \mathcal{U}_\ell)$, the security of aFE implies $\overline{\mathsf{G}}_0^{(\gamma)} \approx_c \overline{\mathsf{G}}_1^{(\gamma)}$.

Game $\overline{\mathsf{G}}_{2}^{(\gamma)}$ for $\gamma \in \{0, 1\}$: This is the same as $\overline{\mathsf{G}}_{1}^{(\gamma)}$ except that, for each query $\mathcal{O}\mathsf{Enc}(i_{0}, m_{i_{0}}^{(0)}, m_{i_{0}}^{(1)})$ with $m_{i_{0}, \mathrm{pub}} = (\mathbf{y}_{i_{0}}, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$, the challenger runs

$$\widetilde{C}_{i_0} \leftarrow \boxed{\mathsf{Sim}(1^{\lambda}, 1^{|C_{\mathcal{U}_M, i}[\mathsf{idDK}_{i_0}, c_{i_0}^{(n-1)}]|}, 1^{|(\mathsf{idDK}_{i_0}, m_{i_0, \mathrm{pri}}^{(0)})|})}$$

instead of $\widetilde{C}_{i_0} \leftarrow \mathsf{Obf}(1^{\lambda}, C_{\mathcal{U}_M, i}[\mathsf{idDK}_{i_0}, c_{i_0}^{(n-1)}], (\mathsf{idDK}_{i_0}, m_{i_0, \mathrm{pri}}^{(b')}), \sigma_{i_0})$, where b' = b if $\beta = \gamma = 0$ and b' = 0 otherwise. It follows $\overline{\mathsf{G}}_1^{(\gamma)} \approx_c \overline{\mathsf{G}}_2^{(\gamma)}$ from the security of LObf.

Game $\overline{\mathsf{G}}_{3}^{(\gamma)}$ for $\gamma \in \{0, 1\}$: This is the same as $\overline{\mathsf{G}}_{2}^{(\gamma)}$ except that, for each query $\mathcal{O}\mathsf{Enc}(i, m_{i}^{(0)}, m_{i}^{(1)})$ with $m_{i, \text{pub}} = (\mathbf{y}_{i}, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$ and $i \in (\mathcal{U}_{\ell} \cap \mathcal{H}) \setminus \{i_{0}\}$, the challenger computes

$$\begin{split} c_i^{(0)} &\leftarrow \mathsf{aEnc}\big(\mathsf{aMPK}_i, \mathbf{y}_i, \left\lfloor \underline{0^{L(\lambda)}} \right\rfloor \big) \\ \forall k \in [|\mathcal{U}_{\ell}| - 1] \colon \quad c_i^{(k)} \leftarrow \mathsf{idEnc}(\mathsf{idMPK}_{i \triangleright k}, (\mathcal{U}_M, \mathsf{tag}), c_i^{(k-1)}) \end{split}$$

instead of $c_i^{(0)} \leftarrow \operatorname{aEnc}(\operatorname{aMPK}_i, \mathbf{y}_i, \sigma_i)$. For each $i \in \mathcal{U}_{\ell} \setminus \{i_0\}$, there exists $k_0 \in [|\mathcal{U}_{\ell}| - 1]$ such that $i \triangleright k_0 = i_0$. From $\overline{\mathsf{G}}_2^{(\gamma)}$, we observe that the adversary never learns $\operatorname{idDK}_{i_0} \leftarrow \operatorname{idKeyGen}(\operatorname{idMSK}_{i_0}, (\mathcal{U}_{\ell}, \operatorname{tag}_{\ell}))$. Thus, the adversary cannot decrypt $c_i^{(k_0)}$ and, in particular, never obtains $c_i^{(0)}$. It follows $\overline{\mathsf{G}}_2^{(\gamma)} \approx_c \overline{\mathsf{G}}_3^{(\gamma)}$ from the security of idFE.

Game $\overline{\mathsf{G}}_{4}^{(\gamma)}$ for $\gamma \in \{0, 1\}$: This is the same as $\overline{\mathsf{G}}_{3}^{(\gamma)}$ except that, for each query $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ with $m_{i, \text{pub}} = (\mathbf{y}_i, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$ and $i \in (\mathcal{U}_{\ell} \cap \mathcal{H}) \setminus \{i_0\}$, the challenger runs

$$\widetilde{C}_i \leftarrow \boxed{\mathsf{Sim}\big(1^{\lambda}, 1^{|C_{\mathcal{U}_M, i}[\mathsf{idDK}_i, c_i^{(n-1)}]|}, 1^{|(\mathsf{idDK}_{i_0}, m_{i_0, \mathrm{pri}}^{(0)})|}\big)}$$

instead of $\widetilde{C}_i \leftarrow \mathsf{Obf}(1^{\lambda}, C_{\mathcal{U}_M, i}[\mathsf{idDK}_i, c_i^{(n-1)}], (\mathsf{idDK}_{i_0}, m_{i_0, \mathrm{pri}}^{(b')}), \sigma_i)$. It follows $\overline{\mathsf{G}}_3^{(\gamma)} \approx_c \overline{\mathsf{G}}_4^{(\gamma)}$ from the security of LObf. Furthermore, we observe that the replies to oracle queries in $\overline{\mathsf{G}}_4^{(\gamma)}$ are independent of $\gamma \in \{0, 1\}$. Therefore, we have $\overline{\mathsf{G}}_4^{(0)} \equiv \overline{\mathsf{G}}_4^{(1)}$ which concludes the proof of the claim.

5.2 DDFE for AB-AWS Secure Against Any Queries

In this section, we present a simple conversion that lifts a DDFE for AB-AWS with legitimatequery security to any-query security, under the condition that the tag space for function keys has polynomial size. Construction 37 (DDFE for AB-AWS Secure Against Any Queries). The construction uses the following ingredients:

- A DDFE wmFE = (wmGSetup, wmLSetup, wmKeyGen, wmEnc, wmDec) for AB-AWS secure against legitimate queries.
- A DDFE anFE = (anSetup, anKeyGen, anEnc, anDec) for AB-AoNE.

The details of the DDFE scheme FE for AB-AWS secure against any queries go as follows:

GSetup (1^{λ}) : On input the security parameter 1^{λ} , run wmPP \leftarrow wmGSetup (1^{λ}) , anPP \leftarrow anGSetup (1^{λ}) and output PP := (wmPP, anPP).

LSetup(PP, i): On input PP and a user $i \in ID$, generate

 $\{(\mathsf{wmSK}_{i,\mathsf{tag-f}},\mathsf{wmPK}_{i,\mathsf{tag-f}}) \leftarrow \mathsf{wmLSetup}(\mathsf{wmPP})\}_{\mathsf{tag-f}\in\mathsf{Tag-f}} \\ \{(\mathsf{anSK}_{i,\mathsf{tag-f}},\mathsf{anPK}_{i,\mathsf{tag-f}}) \leftarrow \mathsf{anLSetup}(\mathsf{anPP})\}_{\mathsf{tag-f}\in\mathsf{Tag-f}}$

and output (SK_i := {wmSK_{i,tag-f}, anSK_{i,tag-f}}_{tag-f} (FTag-f, PK_i := {wmPK_{i,tag-f}, anPK_{i,tag-f}}_{tag-f}). KeyGen(SK_i, k_i): On input SK_i and k_i = (k_{i,pri}, k_{i,pub}), parse k_{i,pub} = (g_i, h_i, U_K, tag-f), compute and output DK_i := (wmDK_{i,tag-f}, anDK_{i,tag-f}) as follows:

$$\begin{split} \mathsf{wmDK}_{i,\mathsf{tag-f}} &\leftarrow \mathsf{wmKeyGen}(\mathsf{wmSK}_{i,\mathsf{tag-f}},k_i) \\ \mathsf{anDK}_{i,\mathsf{tag-f}} &\leftarrow \mathsf{anKeyGen}(\mathsf{anSK}_{i,\mathsf{tag-f}},k_i' = (\top,(g_i,\mathcal{U}_K))) \end{split}$$

Enc(SK_i, m_i): On input SK_i and m_i = (m_{i,pri}, m_{i,pub}), parse m_{i,pub} = ($\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_{j \in [N_i]}, \mathcal{U}_M, \mathsf{tag}$) and output CT_i := {anCT_{i,tag-f}}_{tag-f ∈ Tag-f} as follows:

$$\begin{split} \mathsf{wmCT}_{i,\mathsf{tag-f}} &\leftarrow \mathsf{wmEnc}(\mathsf{wmSK}_{i,\mathsf{tag-f}}, m_i) \\ m'_{i,\mathsf{tag-f}} &\coloneqq (m'_{i,\mathsf{tag-f},\mathrm{pri}} = \mathsf{wmCT}_{i,\mathsf{tag-f}}, m'_{i,\mathsf{tag-f},\mathrm{pub}} = (\mathbf{y}_i, \mathcal{U}_M, \mathsf{tag})) \\ \mathsf{anCT}_{i,\mathsf{tag-f}} &\leftarrow \mathsf{anEnc}(\mathsf{anSK}_{i,\mathsf{tag-f}}, m'_{i,\mathsf{tag-f}}) \end{split}$$

 $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in\mathcal{U}_K},\{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}): On input a set of secret keys \{\mathsf{DK}_i\}_{i\in\mathcal{U}_K} and a set of ciphertexts \\ \{\mathsf{CT}_i\}_{i\in\mathcal{U}_M}, if\mathcal{U}_K \neq \mathcal{U}_M output \bot. Otherwise, define \mathcal{U} \coloneqq \mathcal{U}_K, parse \mathsf{DK}_i = (\mathsf{wmDK}_{i,\mathsf{tag-f}},\mathsf{anDK}_{i,\mathsf{tag-f}}) \\ and \mathsf{CT}_i = \{\mathsf{anCT}_{i,\mathsf{tag-f}}\}_{\mathsf{tag-f}\in\mathsf{Tag-f}} for all i \in \mathcal{U} and output d computed as follows: \end{cases}$

$$\begin{aligned} \{\mathsf{wmCT}_{i,\mathsf{tag-f}}\}_{i\in\mathcal{U}} \leftarrow \mathsf{anDec}(\{\mathsf{anDK}_{i,\mathsf{tag-f}}\}_{i\in\mathcal{U}},\{\mathsf{anCT}_{i,\mathsf{tag-f}}\}_{i\in\mathcal{U}}) \\ d \leftarrow \mathsf{wmDec}(\{\mathsf{wmDK}_{i,\mathsf{tag-f}}\}_{i\in\mathcal{U}},\{\mathsf{wmCT}_{i,\mathsf{tag-f}}\}_{i\in\mathcal{U}}) \end{aligned}$$

Correctness and Security. Correctness follows immediately from the correctness of wmFE and anFE. The scheme is efficient if $|Tag-f| = poly(\lambda)$. Security is stated in the following proposition.

Proposition 38. Let $yyy \in \{sel, sadap, adap\}$ and $zzz \in \{sym, asym\}$. If wmFE is stat-yyy-zzz-secure against legitimate queries and anFE is stat-yyy-zzz-secure, then the DDFE in Construction 37 is stat-yyy-zzz-secure against any queries.

Proof. Let q_M denote the number of distinct tuples $(\mathcal{U}, \mathsf{tag}) \in 2^{\mathsf{ID}} \times \mathsf{Tag}$ such that there is an encryption query $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ with public input $m_{i, \mathsf{pub}} = (\mathfrak{R}, \mathcal{U}, \mathsf{tag})$. We denote these tuples by $(\mathcal{U}_1, \mathsf{tag}_1), \ldots, (\mathcal{U}_{q_M}, \mathsf{tag}_{q_M})$ in the order of their first appearance. Furthermore, we fix any order on Tag-f and parse Tag-f = {tag-f_1, \ldots, tag-f_{q_K}}, where q_K denotes the size of Tag-f. We consider hybrid games $\mathsf{G}_{\ell,\kappa}$, for $\ell \in [q_M]$, $\kappa \in [0; q_K]$ and $b \in \{0, 1\}$, where $\mathsf{G}_{\ell,\kappa}^{(b)}$ is the same as the experiment

 $\mathbf{Exp}_{\mathsf{FE},f^{\mathsf{ab-aws}},\mathcal{A}}^{\mathsf{ddfe-b}}(1^{\lambda}) \text{ except that, for the reply to an encryption query } \mathcal{O}\mathsf{Enc}(i,m_i^{(0)},m_i^{(1)}) \text{ with public input } m_{i,\mathrm{pub}} = (\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_j, \mathcal{U}_{\ell'}, \mathsf{tag}_{\ell'}), \text{ the challenger computes for all } \kappa' \in [q_K]$

$$\mathsf{wmCT}_{i,\mathsf{tag-f}_{\kappa'}} \leftarrow \begin{cases} \mathsf{wmEnc}(\mathsf{wmSK}_{i,\mathsf{tag-f}_{\kappa'}}, \boxed{m_i^{(0)}}) & \text{if } \ell' < \ell \lor (\ell = \ell' \land \kappa' \le \kappa) \\ \mathsf{wmEnc}(\mathsf{wmSK}_{i,\mathsf{tag-f}_{\kappa'}}, m_i^{(b)}) & \text{if } \ell' > \ell \lor (\ell = \ell' \land \kappa' > \kappa) \end{cases},$$

and sends $\mathsf{CT}_i \coloneqq \{\mathsf{anCT}_{i,\mathsf{tag-f}_{\kappa'}}\}_{\kappa' \in [q_K]}$ to \mathcal{A} computed as

$$\begin{split} m'_{i, \mathsf{tag-f}_{\kappa'}} &\coloneqq (m'_{i, \mathsf{tag-f}_{\kappa'}, \mathrm{pri}} = \mathsf{wmCT}_{i, \mathsf{tag-f}_{\kappa'}}, m'_{i, \mathsf{tag-f}_{\kappa'}, \mathrm{pub}} = (\mathbf{y}_i, \mathcal{U}_\ell, \mathsf{tag}_\ell))\\ \mathsf{anCT}_{i, \mathsf{tag-f}_{\kappa'}} &\leftarrow \mathsf{anEnc}(\mathsf{anSK}_{i, \mathsf{tag-f}_{\kappa'}}, m'_{i, \mathsf{tag-f}_{\kappa'}}) \end{split}$$

We can observe that, for $b \in \{0, 1\}$, $\mathsf{G}_{0,0}^b = \mathbf{Exp}_{\mathsf{FE}, f^{\mathsf{ab-aws}}, \mathcal{A}}^{\mathsf{ddfe}, b}(1^\lambda)$, and $\mathsf{G}_{q_M, q_K}^{(0)} \equiv \mathsf{G}_{q_M, q_K}^{(1)}$. Furthermore, we have $\mathsf{G}_{\ell, q_K}^b = \mathsf{G}_{\ell+1,0}^b$ for all $\ell \in [q_M - 1]$. Thus, all we need to prove is the following claim for all $\ell \in [q_M]$, $\kappa \in [q_K]$ and $b \in \{0, 1\}$.

Claim 39. If an FE and wm FE are secure, then $\mathsf{G}_{\ell,\kappa-1}^{(b)} \approx_c \mathsf{G}_{\ell,\kappa}^{(b)}$.

This concludes the proof of the proposition.

We now turn to the proof of the claim.

Proof (of Claim 39). We define an event $E_{\ell,\kappa}$ as follows:

$$\begin{split} E_{\ell,\kappa}: \mbox{ For all } i \in \mathcal{U}_{\ell} \setminus \mathcal{C}, \mbox{ there exist } (i,k_i^{(0)},k_i^{(1)}) \in \mathcal{Q}_{\rm key} \mbox{ with } k_{i,{\rm pub}} = \\ (g_i,h_i,\mathcal{U}_{\ell},{\tt tag-f}_{\kappa}) \mbox{ and } (i,m_i^{(0)},m_i^{(1)}) \in \mathcal{Q}_{\rm enc} \mbox{ with } m_{i,{\rm pub}} = \\ (\mathbf{y}_i,\{\mathbf{x}_{i,j}\}_j,\mathcal{U}_{\ell},{\tt tag}_{\ell}) \mbox{ such that } g_i(\mathbf{y}_i) = 0, \mbox{ or there is no } \\ \mbox{ key generation with a public input of the form } k_{i,{\rm pub}} = \\ (\bigstar,\mathcal{U}_{\ell},{\tt tag-f}_{\kappa}) \mbox{ at all.} \end{split}$$

The proof is done via a sequence of hybrids $\widehat{\mathsf{G}}_0^{\beta}, \ldots, \widehat{\mathsf{G}}_3^{\beta}$ for $\beta \in \{0, 1\}$.

Game $\widehat{\mathsf{G}}_{0}^{(\beta)}$ for $\beta \in \{0,1\}$ This is game $\mathsf{G}_{\ell,\kappa-1+\beta}^{(b)}$.

- **Game** $\widehat{\mathsf{G}}_{1}^{(\beta)}$ for $\beta \in \{0, 1\}$ This is the same as $\mathsf{G}_{0}^{(\beta)}$ except that the challenger chooses a random bit $d \stackrel{\text{\tiny{e}}}{\leftarrow} \{0, 1\}$ during Initialize. Upon \mathcal{A} calling Finalize, if $[d = 0 \text{ and } E_{\ell} \text{ occurs}]$ or $[d = 1 \text{ and } E_{\ell} \text{ does not occur}]$, the simulator outputs 0. We have $\widehat{\mathsf{G}}_{0}^{(\beta)} \approx_{s} \widehat{\mathsf{G}}_{1}^{(\beta)}$.
- **Game** $\widehat{\mathsf{G}}_{2}^{(\beta)}$ for $\beta \in \{0, 1\}$ This game is the same as $\widehat{\mathsf{G}}_{1}^{(\beta)}$ except that if d = 1, then the challenger computes

$$\mathsf{wmCT}_{i,\mathsf{tag-f}_{\kappa}} \gets \mathsf{wmEnc}(\mathsf{wmSK}_{i,\mathsf{tag-f}_{\kappa}}, \boxed{m_i^{(0)}})$$

for the reply to an encryption query $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ with a public input of the form $m_{i,\text{pub}} = (\bigstar, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell})$. We have $\widehat{\mathsf{G}}_1^{(\beta)} \approx_c \widehat{\mathsf{G}}_2^{(\beta)}$ under the security of wmFE.

Game $\widehat{\mathsf{G}}_{3}^{(\beta)}$ for $\beta \in \{0, 1\}$ This game is the same as $\widehat{\mathsf{G}}_{2}^{(\beta)}$ except that if d = 0, then the challenger computes

$$\begin{split} m'_{i, \mathsf{tag-f}_{\kappa}} &\coloneqq (m'_{i, \mathsf{tag-f}_{\kappa'}, \mathrm{pri}} = \bigsqcup , m'_{i, \mathsf{tag-f}_{\kappa}, \mathrm{pub}} = (\mathbf{y}_i, \mathcal{U}_{\ell}, \mathsf{tag}_{\ell}))\\ \mathsf{anCT}_{i, \mathsf{tag-f}_{\kappa}} &\leftarrow \mathsf{anEnc}(\mathsf{anSK}_{i, \mathsf{tag-f}_{\kappa}}, m'_{i, \mathsf{tag-f}_{\kappa}}) \end{split}$$

for the reply to an encryption query $\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})$ with a public input of the form $m_{i,\text{pub}} = (\bigstar, \mathcal{U}_\ell, \mathsf{tag}_\ell)$. We have $\widehat{\mathsf{G}}_1^{(\beta)} \approx_c \widehat{\mathsf{G}}_2^{(\beta)}$ under the security of anFE. Furthermore, we can observe that $\widehat{\mathsf{G}}_3^{(0)} \equiv \widehat{\mathsf{G}}_3^{(1)}$ which concludes the proof of the claim.

By instantiating wmFE with the DDFE for AB-AWS from Section 4 and anFE with the DDFE for AB-AoNE from Section 5.1, we obtain the following theorem.

Theorem 40. Assuming LWE and SXDH, there exists an stat-sadap-sym-secure DDFE for AB-AWS in the ROM. For efficiency, the function tag space Tag-f and sets $U \subseteq ID$ of users must have polynomial and constant size, respectively.

Acknowledgments

This work was supported in part by the French ANR Project ANR-19-CE39-0011 PRESTO and the France 2030 ANR Project ANR-22-PECY-003 SecureCompute.

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.
- ABF⁺13. Joël Alwen, Manuel Barbosa, Pooya Farshim, Rosario Gennaro, S. Dov Gordon, Stefano Tessaro, and David A. Wilson. On the relationship between functional encryption, obfuscation, and fully homomorphic encryption. In Martijn Stam, editor, 14th IMA International Conference on Cryptography and Coding, volume 8308 of LNCS, pages 65–84. Springer, Heidelberg, December 2013.
- 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.
- ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing innerproduct functional encryption. In Dongdai Lin and Kazue Sako, editors, PKC 2019, Part II, volume 11443 of LNCS, pages 128–157. Springer, Heidelberg, April 2019.
- 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.
- 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.
- AGT21a. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, CRYPTO 2021, Part IV, volume 12828 of LNCS, pages 208–238, Virtual Event, August 2021. Springer, Heidelberg.
- AGT21b. 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 Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 711–740. Springer, Heidelberg, November 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.
- AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, CRYPTO 2015, Part I, volume 9215 of LNCS, pages 308–326. Springer, Heidelberg, August 2015.
- AKPW13. Joël Alwen, Stephan Krenn, Krzysztof Pietrzak, and Daniel Wichs. Learning with rounding, revisited new reduction, properties and applications. In Ran Canetti and Juan A. Garay, editors, CRYPTO 2013, Part I, volume 8042 of LNCS, pages 57–74. Springer, Heidelberg, August 2013.
- ALdP11. Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, PKC 2011, volume 6571 of LNCS, pages 90–108. Springer, Heidelberg, March 2011.
- ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, CRYPTO 2016, Part III, volume 9816 of LNCS, pages 333–362. Springer, Heidelberg, August 2016.
- ARYY23. Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 532–564. Springer, Heidelberg, August 2023.
- AS17. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, EUROCRYPT 2017, Part I, volume 10210 of LNCS, pages 152–181. Springer, Heidelberg, April / May 2017.
- ATY23. Shweta Agrawal, Junichi Tomida, and Anshu Yadav. Attribute-based multi-input FE (and more) for attribute-weighted sums. In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part IV, volume 14084 of LNCS, pages 464–497. Springer, Heidelberg, August 2023.
- AYY22. Shweta Agrawal, Anshu Yadav, and Shota Yamada. Multi-input attribute based encryption and predicate encryption. In Yevgeniy Dodis and Thomas Shrimpton, editors, CRYPTO 2022, Part I, volume 13507 of LNCS, pages 590–621. Springer, Heidelberg, August 2022.

- BBL17. Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In Serge Fehr, editor, PKC 2017, Part II, volume 10175 of LNCS, pages 36–66. Springer, Heidelberg, March 2017.
- BCFG17. Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, CRYPTO 2017, Part I, volume 10401 of LNCS, pages 67–98. Springer, Heidelberg, August 2017.
- BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.
- BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, EUROCRYPT 2012, volume 7237 of LNCS, pages 719–737. Springer, Heidelberg, April 2012.
- BS23. Nir Bitansky and Tomer Solomon. Bootstrapping Homomorphic Encryption via Functional Encryption. In Yael Tauman Kalai, editor, 14th Innovations in Theoretical Computer Science Conference (ITCS 2023), volume 251, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- 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.
- BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, 56th FOCS, pages 171–190. IEEE Computer Society Press, October 2015.
- BV16. Zvika Brakerski and Vinod Vaikuntanathan. Circuit-ABE from LWE: Unbounded attributes and semiadaptive security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 363–384. Springer, Heidelberg, August 2016.
- 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. Multiclient 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.
- CLT18. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, ASIACRYPT 2018, Part II, volume 11273 of LNCS, pages 733–764. Springer, Heidelberg, December 2018.
- Coc01. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, 8th IMA International Conference on Cryptography and Coding, volume 2260 of LNCS, pages 360–363. Springer, Heidelberg, December 2001.
- Dat20. Pratish Datta. Constrained pseudorandom functions from functional encryption. *Theoretical Computer Science*, 2020.
- 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.
- dPP22. Paola de Perthuis and David Pointcheval. Two-client inner-product functional encryption with an application to money-laundering detection. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, ACM CCS 2022, pages 725–737. ACM Press, November 2022.
- EHK⁺13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, CRYPTO 2013, Part II, volume 8043 of LNCS, pages 129–147. Springer, Heidelberg, August 2013.
- FFMV23. Danilo Francati, Daniele Friolo, Giulio Malavolta, and Daniele Venturi. Multi-key and multi-input predicate encryption from learning with errors. In Carmit Hazay and Martijn Stam, editors, EUROCRYPT 2023, Part III, volume 14006 of LNCS, pages 573–604. Springer, Heidelberg, April 2023.
- FHKP13. Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, February / March 2013.
- FWW23. Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, CRYPTO 2023, Part IV, volume 14084 of LNCS, pages 498–531. Springer, Heidelberg, August 2023.
- Gay20. Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, PKC 2020, Part I, volume 12110 of LNCS, pages 95–120. Springer, Heidelberg, May 2020.

- GGG⁺14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Heidelberg, May 2014.
- GKL⁺13. S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. https://eprint.iacr.org/2013/774.
- GKVW20. Rishab Goyal, Venkata Koppula, Satyanarayana Vusirikala, and Brent Waters. On perfect correctness in (lockable) obfuscation. In Rafael Pass and Krzysztof Pietrzak, editors, TCC 2020, Part I, volume 12550 of LNCS, pages 229–259. Springer, Heidelberg, November 2020.
- GKW17. Rishab Goyal, Venkata Koppula, and Brent Waters. Lockable obfuscation. In Chris Umans, editor, 58th FOCS, pages 612–621. IEEE Computer Society Press, October 2017.
- 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.
- GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, 40th ACM STOC, pages 197–206. ACM Press, May 2008.
- 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.
- Lan23. Roman Langrehr. On the multi-user security of lwe-based nike, 2023. https://eprint.iacr.org/2023/1401.
- Lin17. Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, CRYPTO 2017, Part I, volume 10401 of LNCS, pages 599–629. Springer, Heidelberg, August 2017.
- LL20. 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.
- LT19. Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 520–551. Springer, Heidelberg, December 2019.
- Ngu24. Duy Nguyen. Dynamic decentralized functional encryptions from pairings in the standard model. Cryptology ePrint Archive, Paper 2024/580, 2024. https://eprint.iacr.org/2024/580.
- 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, Part I, volume 13791 of LNCS, pages 95–125. Springer, Heidelberg, December 2022.
- NPP23. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Optimal security notion for decentralized multiclient functional encryption. In Mehdi Tibouchi and Xiaofeng Wang, editors, ACNS 23, Part II, volume 13906 of LNCS, pages 336–365. Springer, Heidelberg, June 2023.
- NPP24. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with public inputs and strong security. Cryptology ePrint Archive, Paper 2024/740, 2024. https://eprint.iacr. org/2024/740.
- NPS24. Ky Nguyen, David Pointcheval, and Robert Schädlich. Decentralized multi-client functional encryption with strong security. Cryptology ePrint Archive, Report 2024/764, 2024. https://eprint.iacr.org/2024/764.
- NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In 38th FOCS, pages 458–467. IEEE Computer Society Press, October 1997.
- OSW07. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, ACM CCS 2007, pages 195–203. ACM Press, October 2007.
- OT12. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, December 2012.
- QLH⁺24. Xinyuan Qian, Hongwei Li, Meng Hao, Guowen Xu, Haoyong Wang, and Yuguang Fang. Decentralized multi-client functional encryption for inner product with applications to federated learning. *IEEE Transactions on Dependable and Secure Computing*, 2024.
- Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

- SV23. Elaine Shi and Nikhil Vanjani. Multi-Client Inner Product Encryption: Function-Hiding Instantiations Without Random Oracles. In International Conference on Practice and Theory of Public-Key Cryptography (PKC), 2023. https://eprint.iacr.org/2023/615.
- SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, EURO-CRYPT 2005, volume 3494 of LNCS, pages 457–473. Springer, Heidelberg, May 2005.
- Wee21. Hoeteck Wee. Broadcast encryption with size $N^{1/3}$ and more from k-lin. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 155–178, Virtual Event, August 2021. Springer, Heidelberg.
- Wee22. Hoeteck Wee. Optimal broadcast encryption and CP-ABE from evasive lattice assumptions. In Orr Dunkelman and Stefan Dziembowski, editors, EUROCRYPT 2022, Part II, volume 13276 of LNCS, pages 217–241. Springer, Heidelberg, May / June 2022.
- WZ17. Daniel Wichs and Giorgos Zirdelis. Obfuscating compute-and-compare programs under LWE. In Chris Umans, editor, 58th FOCS, pages 600–611. IEEE Computer Society Press, October 2017.

A Supporting Materials – Section 4

A.1 Generic Upgrade of the Security Model in [NPS24]

In this section, we recall the generic compiler from [NPS24] to remove the one-challenge and completequeries constraint for the inner-product functionality in both the standard and function-hiding setting.

Definition 41 (Security for f^{ip} and f^{fh-ip}). A DMCFE scheme is said to be

- secure against complete queries if it is secure against all PPT adversaries that satisfy the following additional condition: for all tag, tag-f ∈ Tag, if there exists a tuple of the form (i, tag-f, *, *) ∈ Q_{key} (resp. (i, tag, *, *) ∈ Q_{enc}) for some i ∈ [n], then there exist tuples (j, tag-f, *, *) ∈ Q_{key} (resp. (j, tag, *, *) ∈ Q_{enc}) for all j ∈ [n],
- one-challenge secure if it is secure against all PPT adversary that declare two additional "challenge" tags tag*, tag-f* ∈ Tag up front to Initialize such that for all tag, tag-f ∈ Tag:
 if (i, tag-f, k_i⁽⁰⁾, k_i⁽¹⁾) ∈ Q_{key} and tag-f ≠ tag-f*, then k_i⁽⁰⁾ = k_i⁽¹⁾.
 if (i, tag, m_i⁽⁰⁾, m_i⁽¹⁾) ∈ Q_{enc} and tag ≠ tag*, then m_i⁽⁰⁾ = m_i⁽¹⁾,

The compiler of [NPS24] uses a DMCFE scheme for a functionality called *All-or-Nothing Encapsulation* (AoNE).

Definition 42 (All-or-Nothing Encapsulation). For $\lambda \in \mathbb{N}$, let $\mathsf{Tag}_{\lambda} = \mathcal{R}_{\lambda} = \{0, 1\}^{\mathrm{poly}(\lambda)}$, $\mathcal{K}_{\lambda} = \emptyset$, $\mathcal{M}_{\lambda,\mathrm{pub}} = \mathsf{Tag}^{13}$ and $\mathcal{M}_{\lambda,\mathrm{pri}} = \{0, 1\}^{L}$ for a polynomial $L = L(\lambda) \colon \mathbb{N} \to \mathbb{N}$. The all-or-nothing encapsulation functionality $f^{\mathsf{aone}} = \{f_{\lambda,n}^{\mathsf{aone}}\}_{\lambda,n\in\mathbb{N}}$ is defined via

$$f_{\lambda,n}^{\mathsf{aone}}(\varnothing, \{m_i\}_{i \in [n]}) = \begin{cases} \{m_{i, \mathrm{pri}}\}_{i \in [n]} & \text{if condition (*) holds} \\ \bot & \text{otherwise} \end{cases}$$

for all $\lambda, n \in \mathbb{N}$, where condition (*) holds if there exists $tag \in Tag_{\lambda}$ such that for each $i \in [n]$, m_i is of the form $(m_{i,pri}, m_{i,pub} := tag)$.

As $\mathcal{K}_{\lambda} = \emptyset$, there is no concept of keys, thus KeyGen is unnecessary and Dec works without taking secret keys as input. Recall from Definition 12 that the functionality f^{ip} is defined w.r.t. $\mathcal{K}_{\lambda,pri} = \mathcal{M}_{\lambda,pri} = [-B; B]^N$ for polynomials $B = B(\lambda)$ and $N = N(\lambda) \colon \mathbb{N} \to \mathbb{N}$. Being more specific, we will write $f_{B,N}^{ip}$ for this functionality below.

We recall the generic compiler from [NPS24] to remove the complete-queries constraint for the inner-product functionality f^{ip} .

Construction 43 ([NPS24]). The construction uses the following ingredients:

- A DMCFE scheme cFE = (cSetup, cKeyGen, cEnc, cDec) for the functionality $f_{B,2N}^{ip}$ that is one-challenge secure against complete queries.
- A DMCFE scheme aFE = (aSetup, aEnc, aDec) for the functionality f^{aone}.

The details of the DMCFE FE = (Setup, KeyGen, Enc, Dec) for $f_{B,N}^{\text{fh-ip}}$ go as follows:

¹³ Slightly abusing notation, we view Tag as part of $\mathcal{M}_{\lambda} = \mathcal{M}_{\lambda,\text{pri}} \times \mathcal{M}_{\lambda,\text{pub}}$, even though in the syntax of DMCFE the tag tag \in Tag_{λ} is a separate input to the encryption algorithm Enc(SK_i, tag, m_i) rather than part of $m_i \in \mathcal{M}_{\lambda}$.

Setup $(1^{\lambda}, 1^{n})$: On input the security parameter 1^{λ} and the support 1^{n} , run

 $(\mathsf{cPP}, \{\mathsf{cSK}_i\}_{i \in [n]}) \leftarrow \mathsf{cSetup}(1^{\lambda}, 1^n)$ $(\mathsf{aPP}, \{\mathsf{aSK}_i\}_{i \in [n]}) \leftarrow \mathsf{aSetup}(1^{\lambda}, 1^n)$

and return $PP \coloneqq (cPP, aPP), \{SK_i \coloneqq (cSK_i, aSK_i)\}_{i \in [n]}.$ KeyGen $(SK_i, tag-f, k_i)$: On input a secret key $SK_i = (cSK_i, aSK_i), a \text{ tag tag-f}, and k_i = (k_{i,pri} = \mathbf{y}_i, k_{i,pri} = \top), define k_i = (k'_{i,pri} = (\mathbf{y}_i \parallel 0^N), k'_{i,pri} = \top) and output DK_i computed as follows:$

 $\begin{aligned} \mathsf{cDK}_i \leftarrow \mathsf{cKeyGen}(\mathsf{cSK}_i, \mathsf{tag-f}, k'_i) \\ \mathsf{DK}_i \leftarrow \mathsf{aEnc}(\mathsf{aSK}_i, \mathsf{tag-f}, \mathsf{cDK}_i) \end{aligned}$

Enc(EK_i, tag, m_i): On input a secret key SK_i = (cSK_i, aSK_i), a tag tag, and $m_i = (m_{i,\text{pri}} = \mathbf{x}_i, m_{i,\text{pri}} = \top)$, define $m_i = (m'_{i,\text{pri}} = (\mathbf{x}_i \parallel 0^N), m'_{i,\text{pri}} = \top)$ and output CT_i computed as follows:

$$cCT_i \leftarrow cEnc(cSK_i, tag, m'_i)$$
$$CT_i \leftarrow aEnc(aSK_i, tag, cCT_i)$$

 $Dec(\{DK_i\}_{i \in [n]}, \{CT_i\}_{i \in [n]})$: On input a set of functional keys $\{DK_i\}_{i \in [n]}$ and a set of ciphertexts $\{CT_i\}_{i \in [n]}$, compute

$$\begin{aligned} \{\mathsf{cDK}_i\}_{i\in[n]} &\leftarrow \mathsf{aDec}(\{\mathsf{DK}_i\}_{i\in[n]}) \\ \{\mathsf{cCT}_i\}_{i\in[n]} &\leftarrow \mathsf{aDec}(\{\mathsf{aCT}_i\}_{i\in[n]}) \end{aligned}$$

If one of these decryption processes returns \perp , output \perp . Otherwise, output

 $d \leftarrow \mathsf{cDec}(\{\mathsf{cDK}_i\}_{i \in [n]}, \{\mathsf{cCT}_i\}_{i \in [n]})$.

The compiler for the functionality $f^{\text{fh-ip}}$ is exactly the same except that the vectors \mathbf{y}_i are part of $k_{i,\text{pri}}$ instead of $k_{i,\text{pub}}$ for $i \in [n]$.

Proposition 44 ([NPS24], Lemmas 2, 3 and 4). Let $xxx \in \{stat, dyn\}$, $yyy \in \{sel, adap\}$, $zzz \in \{sym, asym\}$. If cFE is one-challenge xxx-yyy-zzz-secure against complete queries, then FE is xxx-yyy-zzz-secure against all PPT adversaries.

We next argue that Construction 43 preserves the dynamizability property.

Lemma 45 (Dynamizability). If cFE is \mathbb{A} -dynamizable and aFE is \mathbb{B} -dynamizable for Abelian groups \mathbb{A} and \mathbb{B} , then FE is $\mathbb{A} \times \mathbb{B}$ dynamizable.

Proof. We assume that cFE and aFE are dynamizable, so they are equipped with additional algorithms cSetupPP, cSetupUser, aSetupPP and aSetupUser. Let $(a_i, b_i)_{i \in [n]} \in S(n, \mathbb{A} \times \mathbb{B})$. Then FE admits the following implementation of the algorithms SetupPP and SetupUser.

SetupPP(1^{λ}): Run cPP \leftarrow cSetupPP(1^{λ}) and aPP \leftarrow aSetupPP(1^{λ}), then output PP := (cPP, aPP). SetupUser(PP, (a_i, b_i)): Compute cSK_i \leftarrow cSetupUser(cPP, a_i) as well as aSK_i \leftarrow aSetupUser(aPP, b_i), then output SK_i := (cSK_i, aSK_i).

Then the distributions

$$\left\{ \mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]} \; \left| \begin{array}{c} (a_i, b_i)_{i \in [n]} \xleftarrow{\hspace{1.5pt} \$} \mathcal{S}(n, \mathbb{A} \times \mathbb{B}) \\ \mathsf{PP} \leftarrow \mathsf{Setup}\mathsf{PP}(1^\lambda) \\ \forall i \in [n] \colon \mathsf{SK}_i \leftarrow \mathsf{SetupUser}(\mathsf{PP}, (a_i, b_i)) \end{array} \right\}$$

and $\{(\mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)\}$ are identical.

In [CDSG⁺20], the authors give two possible instantiations for aFE that are \mathbb{O} -dynamizable, where \mathbb{O} denotes the trivial group with one element. In this case, we obtain that FE is $\mathbb{A} \times \mathbb{O}$ -dynamizable, and $\mathbb{A} \times \mathbb{O}$ and \mathbb{A} are isomorphic. The construction in [CDSG⁺20, Section 4] uses solely IBE, thus can be based on both SXDH [BF01] or LWE [GPV08]. Combining Propositions 44 and Lemma 45, we obtain the following theorem.

Theorem 46. Let $xxx \in \{\text{stat}, \text{dyn}\}$, $yyy \in \{\text{sel}, \text{adap}\}$, $zzz \in \{\text{sym}, \text{asym}\}$ and \mathbb{A} be an Abelian group. If there exists an \mathbb{A} -dynamizable DMCFE for f^{ip} or fh-ip that is one-challenge xxx-yyy-zzz-secure against complete queries, then there exists an \mathbb{A} -dynamizable xxx-yyy-zzz-secure DMCFE.

A.2 Instantiation of Construction 25 with a DMCFE for AB-AWS

The function class AB-AWS is defined in Definition 13. Our construction follows the blueprint of [NPS24]. In Section 2, they give an informal description of a compiler from function-hiding single-client IPFE to function-hiding DMCFE for inner products. We follow their blueprint but replace the single-client FE scheme for inner products with one for the AB-AWSw/IP functionality.

Construction 47 (DMCFE Scheme for Attribute-based AWS). The construction uses an *FE scheme* aFE = (aSetup, aKeyGen, aEnc, aDec) for the *AB-AWSw/IP* functionality based on a pairing group $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, q)$. The details of the DMCFE scheme FE for the *AB-AWS* functionality go as follows:

- Setup $(1^{\lambda}, 1^{n})$: On input the security parameter 1^{λ} and the number of clients 1^{n} , sample two fulldomain hash functions H_{1} and H_{2} onto \mathbb{G}_{1}^{2} and \mathbb{G}_{2}^{2} , respectively. Furthermore, run $\mathsf{aMSK}_{i} \leftarrow \mathsf{aSetup}(1^{\lambda})$ for each $i \in [n]$ and sample $s_{1}, \ldots, s_{n} \notin \mathbb{Z}_{q}$ conditioned on $\sum_{i \in [n]} s_{i} = 0$. Output $\mathsf{PP} \coloneqq (\mathsf{H}_{1}, \mathsf{H}_{2})$ and $\{\mathsf{SK}_{i} \coloneqq (s_{i}, \mathsf{aMSK}_{i})\}_{i \in [n]}$.
- $\mathsf{KeyGen}(\mathsf{SK}_i, \mathsf{tag-f}, k_i): On \ input \ a \ secret \ key \ \mathsf{SK}_i = (s_i, \mathsf{aMSK}_i), \ a \ tag \ \mathsf{tag-f} \ and \ k_i = (k_{i, \mathrm{pri}} = \top, k_{i, \mathrm{pub}} = (g_i, h_i)), \ compute \ [\![\tau_{\mathsf{tag-f}}]\!]_2 \leftarrow \mathsf{H}_2(\mathsf{tag-f}) \ and \ output \ \mathsf{DK}_i \coloneqq \mathsf{aDK}_i \ as \ follows:$

$$k'_i \coloneqq (k'_{i,\text{pri}} = \llbracket (\tau_{\text{tag-f}}, 0) \rrbracket_2, k'_{i,\text{pub}} = k_{i,\text{pub}}) \quad \text{aDK}_i \leftarrow \mathsf{aKeyGen}(\mathsf{aMSK}_i, k')$$

 $\mathsf{Enc}(\mathsf{SK}_i, \mathsf{tag}, m_i): \text{ On input a secret key } \mathsf{SK}_i = (s_i, \mathsf{aMSK}_i), \text{ a tag tag and } m_i = (m_{i, \mathrm{pri}} = \{\mathbf{z}_{i, j}\}_{j \in [N_i]}, m_{i, \mathrm{pub}} = (\mathbf{y}_i, \{\mathbf{x}_i\}_{j \in [N_i]})), \text{ compute } [\![\sigma_{\mathsf{tag}}]\!]_1 \leftarrow \mathsf{H}_1(\mathsf{tag}) \text{ and output } \mathsf{CT}_i \coloneqq \mathsf{aCT}_i \text{ as follows:}$

$$\begin{split} m'_i &\coloneqq \left(m'_{i,\text{pri}} = (m_{i,\text{pri}}, \llbracket (s_i \sigma_{\text{tag}}, 0) \rrbracket_1), m'_{i,\text{pub}} = m_{i,\text{pub}} \right) \\ \mathsf{aCT}_i &\leftarrow \mathsf{aEnc}(\mathsf{aMSK}_i, m'_i) \end{split}$$

 $\begin{aligned} \mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in[n]},\{\mathsf{CT}_i\}_{i\in[n]}): & On \text{ input a set of decryption keys } \{\mathsf{DK}_i=\mathsf{aDK}_i\}_{i\in[n]} \text{ and a set of } \\ & \text{ciphertexts } \{\mathsf{CT}_i=\mathsf{aCT}_i\}_{i\in[n]}, \text{ compute } \rho_i \leftarrow \mathsf{aDec}(\mathsf{aDK}_i,\mathsf{aCT}_i) \text{ for all } i\in[n]. \text{ If there exists } i \\ & \text{such that } \rho_i=\bot, \text{ then output } \bot. \text{ Otherwise, parse } [\![d_i]\!]_{\mathsf{t}} = \rho_i \text{ and return } [\![d]\!]_{\mathsf{t}} = \sum_{i\in[n]} [\![d_i]\!]_{\mathsf{t}}. \end{aligned}$

Correctness. For all $i \in [n]$, if $g_i(\mathbf{y}_i) = 0$, then we have

$$d_i = \sum_{j \in [N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle + \langle s_i \sigma_{\mathsf{tag}}, \tau_{\mathsf{tag-f}} \rangle$$

by the correctness of aFE. Hence, we conclude $d = \sum_{i \in [n]} \sum_{j \in [N_i]} \langle f(\mathbf{x}_{i,j}), \mathbf{z}_{i,j} \rangle$ from the fact that $\sum_{i \in [n]} s_i = 0$.

Dynamizability. The following lemma argues that the construction fits into the framework of dynamizable DMCFE schemes.

Lemma 48. The DMCFE scheme FE in Construction 47 is \mathbb{Z}_q -dynamizable.

Proof. Let $(s_i)_{i \in [n]} \in \mathcal{S}(n, \mathbb{Z}_q)$. The scheme admits the following implementation of the algorithms SetupPP and SetupUser.

SetupPP(1^{λ}): Sample two full-domain hash functions H₁ and H₂ onto G₁ and G₂ respectively. Output PP := (H₁, H₂).

SetupUser(PP, s_i): Generate $\mathsf{aMSK}_i \leftarrow \mathsf{aSetup}(1^\lambda)$ and $\mathsf{output} \mathsf{SK}_i \coloneqq (\mathsf{aMSK}_i, s_i)$.

Then the distributions

$$\left\{ \mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]} \middle| \begin{array}{c} (s_i)_{i \in [n]} \stackrel{\text{\tiny{\&}}}{\leftarrow} \mathcal{S}(n, \mathbb{Z}_q) \\ \mathsf{PP} \leftarrow \mathsf{Setup}\mathsf{PP}(1^\lambda) \\ \forall i \in [n] \colon \mathsf{SK}_i \leftarrow \mathsf{SetupUser}(\mathsf{PP}, s_i) \end{array} \right\}$$

and $\{(\mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^n)\}$ are identical.

Security. The following lemma proves security against legitimate queries.

Proposition 49. If aFE is sadap-secure and the SXDH assumption holds in \mathbb{G} , then Construction 47 is stat-sadap-sym-secure against legitimate queries in the random oracle model.

Proof. Let Q be the number of different tags that occur in an encryption query and let $\{ \mathsf{tag}_1, \ldots, \mathsf{tag}_Q \}$ denote the set of these tags with some fixed ordering. We prove the proposition via a series of hybrids $\mathsf{G}_0^{(b)}, \ldots, \mathsf{G}_Q^{(b)}$ where $\mathsf{G}_\ell^{(b)}$, for $\ell \in [0; Q]$ and $b \in \{0, 1\}$, is the same as $\mathsf{Exp}_{\mathsf{FE}, f^{\mathsf{ab-aws}}, \mathcal{A}}^{\mathsf{dmcfe}, b}(1^\lambda)$ except that for encryption queries of the form $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_{\ell'}, m_i^{(0)}, m_i^{(1)})$ with $m_i^{(\beta)} = (m_{i, \mathrm{pri}}^{(\beta)}, m_{i, \mathrm{pub}})$ for $\beta \in \{0, 1\}$, the challenger sets

$$m_{i,\text{pri}}' = \begin{cases} (m_{i,\text{pri}}^{(0)}, \llbracket \mathbf{p}_i \rrbracket_1 = \llbracket (s_i \sigma_{\mathsf{tag}}, 0) \rrbracket_1) & \text{if } \ell' \le \ell \\ (m_{i,\text{pri}}^{(b)}, \llbracket \mathbf{p}_i \rrbracket_1 = \llbracket (s_i \sigma_{\mathsf{tag}}, 0) \rrbracket_1) & \text{if } \ell' > \ell \end{cases}$$

Below, we prove the following claim for all $\ell \in [Q]$:

Claim 50. If aFE is selectively secure and the SXDH assumption holds in \mathbb{G} , then we have $\mathsf{G}_{\ell-1}^{(b)} \approx_c \mathsf{G}_{\ell}^{(b)}$.

Furthermore, we note that $\mathsf{G}_{0}^{(b)} = \mathbf{Exp}_{\mathsf{FE}, f^{\mathsf{ab-aws}}, \mathcal{A}}^{\mathsf{dmcfe}, b}(1^{\lambda})$, for $b \in \{0, 1\}$, and $\mathsf{G}_{Q}^{(0)} \equiv \mathsf{G}_{Q}^{(1)}$ because the adversary's view is independent of the bit b. This concludes the proof of the proposition. \Box

We now prove the claim.

Proof (of Claim 50). We start with a concrete interpretation of the adversary's admissibility condition in the case of the AB-AWS functionality. We denote the input to the κ -th encryption query of the form $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, \star, \star)$ by $(m_{\mathsf{tag},i}^{(\kappa,0)}, m_{\mathsf{tag},i}^{(\kappa,1)})$ with $m_{\mathsf{tag},i}^{(\kappa,\gamma)} = (m_{\mathsf{tag},i,\mathsf{pri}}^{(\kappa,\gamma)} = \{\mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,\gamma)}\}_{j \in [N_{\mathsf{tag},i}^{(\kappa)}]}, m_{\mathsf{tag},i,\mathsf{pub}}^{(\kappa)} = \{\mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,\gamma)}\}_{j \in [N_{\mathsf{tag},i}^{(\kappa)}]}$ $(\mathbf{y}_{\mathsf{tag},i}^{(\kappa)}, {\mathbf{x}_{\mathsf{tag},i,j}^{(\kappa)}}_{j \in [N_{\mathsf{tag},i}^{(\kappa)}]})$ for $\gamma \in \{0,1\}$. Similarly, we denote the public input to the ν -th query of the form $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag-f}, \star, \star)$ by $k^{\nu}_{\mathsf{tag-f}, i, \mathrm{pub}} = (g^{\nu}_{\mathsf{tag-f}, i}, h^{\nu}_{\mathsf{tag-f}, i})^{14}$. The admissibility condition of Definition 11 states that for all tag, tag-f, κ and ν satisfying $g_{\mathsf{tag-f},i}^{\nu}(\mathbf{y}_{\mathsf{tag},i}^{(\kappa)}) = 0$ for all $i \in \mathcal{H}$ where $\mathcal{H} \coloneqq [n] \setminus \mathcal{C}$, we have

$$\sum_{i \in \mathcal{H}} \sum_{j \in [N_{\mathsf{tag},i}^{(\kappa)}]} \left\langle h_{\mathsf{tag-f},i}^{\nu}(\mathbf{x}_{\mathsf{tag},i,j}^{(\kappa)}), \mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,0)} \right\rangle = \sum_{i \in \mathcal{H}} \sum_{j \in [N_{\mathsf{tag},i}^{(\kappa)}]} \left\langle h_{\mathsf{tag-f},i}^{\nu}(\mathbf{x}_{\mathsf{tag},i,j}^{(\kappa)}), \mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,1)} \right\rangle$$

as well as $\mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,0)} = \mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,1)}$ for all $j \in [N_{\mathsf{tag},i}^{(\kappa)}]$ if $i \in \mathcal{C}$. From this, it follows for $\gamma \in \{0,1\}^{15}$ that

$$\Delta_{\mathsf{tag-f},\mathsf{tag},i}^{(\gamma)} \coloneqq \sum_{j \in [N_{\mathsf{tag},i}^{(\kappa)}]} \left\langle h_{\mathsf{tag-f},i}^{\nu}(\mathbf{x}_{\mathsf{tag},i,j}^{(\kappa)}), \mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,\gamma)} - \mathbf{z}_{\mathsf{tag},i,j}^{(\kappa,0)} \right\rangle \tag{1}$$

are constant for all repetitions κ, ν , and $\Delta_{\mathsf{tag-f}, \mathsf{tag}, i}^{(\gamma)} = 0$ if $i \in \mathcal{C}$. Then we also have that $\sum_{i \in \mathcal{H}} \Delta_{\mathsf{tag-f}, \mathsf{tag}, i}^{(\gamma)} = 0$. Together, these conditions imply that the following distributions are identical.

$$D_{0} = \left\{ (s_{i})_{i \in \mathcal{H}} : (s_{i})_{i \in \mathcal{H}} \stackrel{*}{\leftarrow} \mathbb{Z}_{q}^{|\mathcal{H}|} \text{ s.t. } \sum_{i \in \mathcal{H}} s_{i} = 0 \right\}$$

$$D_{1} = \left\{ (s_{i} + \Delta_{\mathsf{tag-f},\mathsf{tag},i}^{(\gamma)})_{i \in \mathcal{H}} : (s_{i})_{i \in \mathcal{H}} \stackrel{*}{\leftarrow} \mathbb{Z}_{q}^{|\mathcal{H}|} \text{ s.t. } \sum_{i \in \mathcal{H}} s_{i} = 0 \right\}$$

$$(2)$$

We consider the following series of hybrids $\widehat{\mathsf{G}}_{0}^{(\beta)}, \ldots, \widehat{\mathsf{G}}_{5}^{(\beta)}$ for $\beta \in \{0, 1\}$.

Game $\widehat{\mathsf{G}}_{0}^{(\beta)}$ for $\beta \in \{0,1\}$: This is game $\mathsf{G}_{\ell-1+\beta}^{(b)}$.

Game $\widehat{\mathsf{G}}_{1}^{(\beta)}$ for $\beta \in \{0,1\}$: This is the same as $\widehat{\mathsf{G}}_{0}^{(\beta)}$ except that the challenger samples random group elements $[\![s_{\mathsf{tag}_{\ell},i}]\!]_1 \stackrel{*}{=} \mathbb{G}_1$ for each $i \in \mathcal{H}$ subject to the condition $\sum_{i \in \mathcal{H}} [\![s_{\mathsf{tag}_{\ell},i}]\!]_1 = -\sum_{i \in \mathcal{C}} s_i \cdot [\![\sigma_{\mathsf{tag}_{\ell}}]\!]_1$. For the reply to an encryption query $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_{\ell}, m_i^{(0)}, m_i^{(1)})$ with respect to tag_{ℓ} and $i \in \mathcal{H}$, the challenger defines

$$m'_{i} \coloneqq \left(m'_{i,\text{pri}} = (m^{(b')}_{i,\text{pri}}, \llbracket \mathbf{p}_{i} \rrbracket_{1} = \llbracket (\ \underline{s_{\mathsf{tag}_{\ell},i}}, 0) \rrbracket_{1}), m'_{i,\text{pub}} = m_{i,\text{pub}} \right) ,$$

where b' = b if $\beta = 0$ and b' = 0 if $\beta = 1$. We have $\widehat{\mathsf{G}}_0^{(\beta)} \approx_c \widehat{\mathsf{G}}_1^{(\beta)}$ under the DDH assumption in \mathbb{G}_1 . Note that we can exploit the random self-reducibility of the DDH problem here, so one DDH instance suffices.

Game $\widehat{\mathsf{G}}_{2}^{(\beta)}$ for $\beta \in \{0, 1\}$: This is the same as $\widehat{\mathsf{G}}_{1}^{(\beta)}$ except that we program H_{1} at the point tag_{ℓ} by sampling $\sigma_{\mathsf{tag}_{\ell}} \stackrel{*}{=} \mathbb{Z}_{q}$ and setting $\mathsf{H}_{1}(\mathsf{tag}_{\ell}) \coloneqq \llbracket \sigma_{\mathsf{tag}_{\ell}} \rrbracket_{1}$. This gives a perfect simulation and we have $\widehat{\mathsf{G}}_{1}^{(\beta)} \equiv \widehat{\mathsf{G}}_{2}^{(\beta)}$.

Game $\widehat{\mathsf{G}}_{3}^{(\beta)}$ for $\beta \in \{0,1\}$: This is the same as $\widehat{\mathsf{G}}_{2}^{(\beta)}$ except that the challenger samples random group elements $[\![s_{\mathsf{tag}}_{\ell}, i]\!]_2 \stackrel{*}{\leftarrow} \mathbb{G}_2$ for each $i \in \mathcal{H}$ subject to the condition $\sum_{i \in \mathcal{H}} [\![s_{\mathsf{tag}}_{\ell}, i]\!]_2 =$

¹⁴ Note that the private key input is always \top in the AB-AWS functionality, so we can ignore it. For this reason, we may also write \mathcal{O} KeyGen $(i, \mathsf{tag-f}, k_{i,\text{pub}})$ instead of \mathcal{O} KeyGen $(i, \mathsf{tag-f}, k_i^{(0)}, k_i^{(1)})$ for brevity. ¹⁵ More precisely, the case $\gamma = 1$ follows from the admissibility condition while for $\gamma = 0$, we always have $\Delta_{\mathsf{tag-f},\mathsf{tag},i}^{(\gamma)} = 0$.

 $-\sum_{i\in\mathcal{C}} s_i \cdot [\![\sigma_{\mathsf{tag}_\ell}]\!]_2$. Note that $[\![\sigma_{\mathsf{tag}_\ell}]\!]_2$ is known thanks to the programming of the random oracle. For the reply to an encryption query $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, m_i^{(0)}, m_i^{(1)})$ with respect to tag_ℓ and $i \in \mathcal{H}$ or a key generation query $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag-f}, k_{i, \text{pub}})$, respectively, the challenger defines

$$\begin{split} m'_{i} &\coloneqq \left(m'_{i,\text{pri}} = (m^{(b')}_{i,\text{pri}}, \llbracket \mathbf{p}_{i} \rrbracket_{1} = \llbracket (\fbox{0,1}) \rrbracket_{1}), m'_{i,\text{pub}} = m_{i,\text{pub}} \right) \\ k'_{i} &\coloneqq \left(k'_{i,\text{pri}} = \llbracket \mathbf{q}_{i} \rrbracket_{2} = \llbracket (\tau_{\text{tag-f}}, \fbox{\tau_{\text{tag-f}} \cdot s_{\text{tag}_{\ell}, i}}) \rrbracket_{2}, k'_{i,\text{pub}} = k_{i,\text{pub}} \right) \ . \end{split}$$

As the inner products between vectors $[\![\mathbf{p}_i]\!]_1$ and $[\![\mathbf{q}_i]\!]_2$ do not change, it follows $\widehat{\mathsf{G}}_2^{(\beta)} \approx_c \widehat{\mathsf{G}}_3^{(\beta)}$ from the security of aFE.

Game $\widehat{\mathsf{G}}_{4}^{(\beta)}$ for $\beta \in \{0, 1\}$: This is the same as $\widehat{\mathsf{G}}_{3}^{(\beta)}$ except that the challenger samples random values $s_{\mathsf{tag-f},\mathsf{tag}_{\ell},i} \stackrel{*}{\leftarrow} \mathbb{Z}_q$ for each $i \in \mathcal{H}$ and $\mathsf{tag-f} \in \mathsf{Tag}$ that occurs in a key generation query subject to the condition $\sum_{i \in \mathcal{H}} s_{\mathsf{tag-f},\mathsf{tag}_{\ell},i} = -\sum_{i \in \mathcal{C}} \tau_{\mathsf{tag-f}} \cdot s_{\mathsf{tag}_{\ell},i}$. For the reply to a query $\mathcal{O}\mathsf{KeyGen}(i,\mathsf{tag-f},k_{i,\mathrm{pub}})$ such that $i \in \mathcal{H}$, the challenger defines

$$k'_i \coloneqq \left(k'_{i,\text{pri}} = \llbracket \mathbf{q}_i \rrbracket_2 = \llbracket (\tau_{\mathsf{tag-f}}, \boxed{s_{\mathsf{tag-f}, \mathsf{tag}_{\ell}, i}}) \rrbracket_2, k'_{i,\text{pub}} = k_{i,\text{pub}} \right)$$

We have $\widehat{\mathsf{G}}_{3}^{(\beta)} \approx_{c} \widehat{\mathsf{G}}_{4}^{(\beta)}$ under the DDH assumption in \mathbb{G}_{2} . Note that we can exploit the random self-reducibility of the DDH problem.

Game $\widehat{\mathsf{G}}_{5}^{(\beta)}$ for $\beta \in \{0,1\}$: This is the same as $\widehat{\mathsf{G}}_{4}^{(\beta)}$ except that, for the reply to an encryption query $\mathsf{aCT}_{\mathsf{tag}_{\ell},i} \leftarrow \mathcal{O}\mathsf{Enc}(i,\mathsf{tag}_{\ell},m_i^{(0)},m_i^{(1)})$ with respect to tag_{ℓ} and $i \in \mathcal{H}$ or a key generation query $\mathsf{aDK}_{\mathsf{tag}-\mathsf{f},i} \leftarrow \mathcal{O}\mathsf{KeyGen}(i,\mathsf{tag}-\mathsf{f},k_{i,\mathrm{pub}})$, respectively, the challenger sets

$$\begin{split} m'_{i} &\coloneqq \left(m'_{i,\text{pri}} = \left(\boxed{m_{i,\text{pri}}^{(0)}}, \llbracket \mathbf{p}_{i} \rrbracket_{1} = \llbracket (0,1) \rrbracket_{1} \right), m'_{i,\text{pub}} = m_{i,\text{pub}} \right) \\ k'_{i} &\coloneqq \left(k'_{i,\text{pri}} = \llbracket \mathbf{q}_{i} \rrbracket_{2} = \llbracket (\tau_{\text{tag-f}}, s_{\text{tag-f},\text{tag}_{\ell}, i} + \boxed{\varDelta_{\text{tag-f},\text{tag}_{\ell}, i}^{(b')}}) \rrbracket_{2}, k'_{i,\text{pub}} = k_{i,\text{pub}} \right) \end{split}$$

We have $\widehat{\mathbf{G}}_{4}^{(\beta)} \approx_{c} \widehat{\mathbf{G}}_{5}^{(\beta)}$ from the security of aFE. This can be seen as follows. Parse the inputs of the queries as $m_{i}^{(\gamma)} = (m_{i,\mathrm{pri}}^{(\gamma)} = \{\mathbf{z}_{i,j}^{(\gamma)}\}_{j\in[N_i]}, m_{i,\mathrm{pub}} = (\mathbf{y}_i, \{\mathbf{x}_{i,j}\}_{j\in[N_i]}))$ for $\gamma \in \{0, 1\}$ and $k_{i,\mathrm{pub}} = (g_i, h_i)$. Let $d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(\kappa)}$ denote the decryption value of $\mathsf{aDec}(\mathsf{aDK}_{\mathsf{tag-f},i}, \mathsf{aCT}_{\mathsf{tag}_{\ell},i})$ in game $\widehat{\mathbf{G}}_{\kappa}^{(\beta)}$ for $\kappa \in \{4, 5\}$. We need to argue that $d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(4)} = d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(5)}$ for all $\mathsf{tag-f}$ and $i \in \mathcal{H}$. For this, we distinguish two cases. If $g_i(\mathbf{y}_{\mathsf{tag}_{\ell}}) \neq 0$, then $d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(4)} = d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(5)} = \bot$. Otherwise, we have $d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(4)} = d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(5)} = \sum_{j\in[N_i]} \langle h_i(\mathbf{x}_{i,j}), \mathbf{z}_{i,j}^{(b)} \rangle$, where in the case $d_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(5)}$ we use the fact that $\Delta_{\mathsf{tag-f},\mathsf{tag}_{\ell},i}^{(b)}$ is a constant for all repetitions, as observed in (1).

Game $\widehat{\mathsf{G}}_{6}^{(\beta)}$ for $\beta \in \{0, 1\}$: This is the same as $\widehat{\mathsf{G}}_{5}^{(\beta)}$ except that, for the reply to a key generation query $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag-}f, k_{i, \text{pub}})$, the challenger sets

$$k'_i \coloneqq \left(k'_{i,\text{pri}} = \llbracket \mathbf{q}_i \rrbracket_2 = \llbracket (\tau_{\mathsf{tag-f}}, s_{\mathsf{tag-f}, \mathsf{tag}_\ell, i} + \overbrace{\varDelta_{\mathsf{tag-f}, \mathsf{tag}_\ell, i}^{(\mathsf{t}')}}) \rrbracket_2, k'_{i,\text{pub}} = k_{i,\text{pub}} \right)$$

As observed in (2), this does not change the distribution of the vector $[\![\mathbf{q}_i]\!]_2$; so we have $\widehat{\mathsf{G}}_5^{(\beta)} \equiv \widehat{\mathsf{G}}_6^{(\beta)}$. Moreover, we observe that in $\widehat{\mathsf{G}}_5^{(\beta)}$ the adversaries view is independent of β which implies $\widehat{\mathsf{G}}_6^{(0)} \equiv \widehat{\mathsf{G}}_6^{(1)}$. This concludes the proof of the claim.

Combining Proposition 49 with the generic conversion from DMCFE to DDFE (Construction 25, Proposition 26), we obtain the following corollary.

Corollary 51. Assuming SXDH in the ROM, there exists a DDFE scheme for f^{ab-aws} that is stat-sel-sym-secure against legitimate queries.

A.3 Instantiation of Construction 25 with the DMCFE of [CDG⁺18a]

We recall the construction of $[CDG^+18a]$ using our notations. Their scheme considers a restricted variant of the inner-product functionality f^{ip} (Definition 12) where each user encrypts sub-vectors of length 1, *i.e.* $N = N(\lambda) = 1$.

Construction 52 (DMCFE Scheme of [**CDG**⁺**18a**]). *The construction is based on a pairing* group $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, q)$. The details of the scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *go as follows:*

Setup(1^λ, 1ⁿ): On input the security parameter 1^λ and the number of clients 1ⁿ sample two full-domain hash functions H₁ and H₂ onto G₁² and G₂², respectively. For each i ∈ [n], generate s_i ^{*} Z_q² and T_i ^{*} Z_q^{2×2} such that ∑_{i∈[n]} T_i = 0. Output (PP := (H₁, H₂), {SK_i := (s_i, T_i)}_{i∈[n]}). KeyGen(SK_i, tag-f, k_i): On input a secret key SK_i = (s_i, T_i), a tag tag-f and k_i = (⊤, y_i), compute [[v]₂ = H₂(tag-f), [[d_i]₂ = [[y_i · s_i + T_i · v]₂ and output DK_i := [[d_i]₂.

Enc(SK_i, tag, m_i): On input a secret key SK_i = ($\mathbf{s}_i, \mathbf{T}_i$), a tag tag and a $m_i = (x_i, \top)$, compute $[\![\mathbf{u}]\!]_1 = \mathsf{H}_1(\mathsf{tag}), [\![c_i]\!]_1 = [\![\langle \mathbf{u}, \mathbf{s}_i \rangle + x_i]\!]_1$ and output CT_i := ($[\![c_i]\!]_1, \mathsf{tag}$).

 $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in[n]}, \{\mathsf{CT}_i\}_{i\in[n]}): On input a set of decryption keys \{\mathsf{DK}_i = \llbracket \mathbf{d}_i \rrbracket_2\}_{i\in[n]} and a set of ciphertexts \{\mathsf{CT}_i = (\llbracket c_i \rrbracket_1, \mathsf{tag}_i)\}_{i\in[n]}, if \mathsf{H}_1(\mathsf{tag}_1) = \cdots = \mathsf{H}_1(\mathsf{tag}_n) =: \llbracket \mathbf{u} \rrbracket_1 compute$

$$[\![\alpha]\!]_t = \sum_{i=1}^n [\![c_i]\!]_1 [\![y_i]\!]_2 - [\![\mathbf{u}]\!]_1 [\![\mathbf{d}_i]\!]_2$$

then find and output the discrete log α . Otherwise, abort with failure.

Proposition 53 ([CDG^+18a]). Construction 52 is stat-adap-sym-secure against complete queries and without repetitions under the SXDH assumption in the ROM.

The following lemma shows that their scheme fits into our framework of dynamizability.

Lemma 54. The DMCFE scheme FE in Construction 52 is $\mathbb{Z}_{q}^{2\times 2}$ -dynamizable.

Proof. Let $(\mathbf{T}_i)_{i \in [n]} \in \mathcal{S}(n, \mathbb{Z}_q^{2 \times 2})$. The scheme admits the following implementation of the algorithms SetupPP and SetupUser.

SetupPP(1^{λ}): Sample two full-domain hash functions H₁ and H₂ onto \mathbb{G}_1^2 and \mathbb{G}_2^2 respectively. Output PP := (H₁, H₂).

SetupUser(PP, \mathbf{T}_i): Sample $\mathbf{s}_i \stackrel{\scriptscriptstyle{\mathfrak{s}}}{\leftarrow} \mathbb{Z}_q^2$ and output $\mathsf{SK}_i \coloneqq (\mathbf{s}_i, \mathbf{T}_i)$.

Then the distributions

$$\begin{cases} \mathsf{PP}, \{\mathsf{SK}_i\}_{i\in[n]} & | (\mathbf{T}_i)_{i\in[n]} \stackrel{*}{\leftarrow} \mathcal{S}(n, \mathbb{Z}_q^{2\times 2}) \\ \mathsf{PP} \leftarrow \mathsf{SetupPP}(1^\lambda) \\ \forall i \in [n] \colon \mathsf{SK}_i \leftarrow \mathsf{SetupUser}(\mathsf{PP}, \mathbf{T}_i) \end{cases} \end{cases}$$

and $\{(\mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^n)\}$ are identical.

Combining Theorem 46 and Proposition 53 with the generic conversion from DMCFE to DDFE (Construction 25, Proposition 26), we obtain the following corollary.

Corollary 55. Assuming SXDH and the ROM, there exists a DDFE scheme for f^{ip} where each user encrypts vectors of length 1 that is stat-adap-sym-secure without repetitions.

A.4 Instantiation of Construction 25 with a Variant of [LT19]

We recall the DMCFE construction of [LT19]. We start with some preliminaries.

Preliminaries. For the preliminaries on lattices, homomorphic encryption, and admissible hash functions, we refer to [LT19, Section 2]. If X and Y are distributions over the same domain \mathcal{D} , then $\Delta(X, Y)$ denotes their statistical distance. Let $\Sigma \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix and $\mathbf{c} \in \mathbb{R}^{\ell}$ be a vector. We define the *Gaussian function* over \mathbb{R}^n by $\rho_{\Sigma, \mathbf{c}}(\mathbf{x}) = \exp(-\pi(\mathbf{x}-\mathbf{c})^\top \Sigma^{-1}(\mathbf{x}-\mathbf{c}))$ and if $\Sigma = \sigma^2 \cdot \mathbf{I}_n$ and $\mathbf{c} = \mathbf{0}$, we write ρ_{σ} for $\rho_{\Sigma, \mathbf{c}}$. For any discrete set $\Lambda \subset \mathbb{R}^n$, the *discrete Gaussian distribution* $D_{\Lambda, \Sigma, \mathbf{c}}$ has probability mass $\Pr_{X \sim D_{\Lambda, \Sigma, \mathbf{c}}}[X = \mathbf{x}] = \rho_{\Sigma, \mathbf{c}}(\mathbf{x})/\rho_{\Sigma, \mathbf{c}}(\Lambda)$, for any $\mathbf{x} \in \Lambda$. When $\mathbf{c} = \mathbf{0}$ and $\Sigma = \sigma^2 \cdot \mathbf{I}_n$ we denote $D_{\Lambda, \Sigma, \mathbf{c}}$ by $D_{\Lambda, \sigma}$.

We will make use of the *Chernoff-Cramér* tail bound for Gaussian random variables. Let $X \sim N(0, \nu)$ where $\nu > 0$ is the variance. Then for any $\beta > 0$, it holds that

$$\Pr[|X| \ge \beta] \le 2 \cdot \exp\left(-\frac{\beta^2}{2\nu}\right) \quad . \tag{3}$$

Furthermore, we need two classical inequalities from calculus. For all x, n > 0, it holds that

$$e^x \le \left(1 + \frac{x}{n}\right)^{n+x/2}$$
 and $1 + x \le e^x$. (4)

Finally, we recall the LWE assumption.

Definition 56 (Learning with Errors). Let $\alpha : \mathbb{N} \to (0,1)$ and $m \ge n \ge 1$, $q \ge 2$ be functions of a security parameter $\lambda \in \mathbb{N}$. We write vectors as column vectors. The Learning with Errors (LWE) problem consists in distinguishing between the distributions $(\mathbf{A}, \mathbf{s}^{\top}\mathbf{A} + \mathbf{e}^{\top})$ and $U(\mathbb{Z}_{q}^{n \times m} \times \mathbb{Z}_{q}^{m})$, where $\mathbf{A} \sim U(\mathbb{Z}_{q}^{n \times m})$, $\mathbf{s} \sim U(\mathbb{Z}_{q}^{n})$ and $\mathbf{e} \sim D_{\mathbb{Z}^{m}, \alpha q}$. For a PPT algorithm $\mathcal{A} : \mathbb{Z}_{q}^{n \times m} \times \mathbb{Z}_{q}^{m} \to \{0, 1\}$, we define

$$\mathbf{Adv}_{q,m,n,\alpha}^{\mathsf{LWE}}(\mathcal{A}) = \left| \Pr[\mathcal{A}\left(\mathbf{A}, \mathbf{s}^{\top}\mathbf{A} + \mathbf{e}^{\top}\right) = 1] - \Pr[\mathcal{A}\left(\mathbf{A}, \mathbf{u}\right) = 1] \right|,$$

where the probabilities are over $\mathbf{A} \sim U(\mathbb{Z}_q^{n \times m})$, $\mathbf{s} \sim U(\mathbb{Z}_q^n)$, $\mathbf{u} \sim U(\mathbb{Z}_q^m)$, $\mathbf{e} \sim D_{\mathbb{Z}^m, \alpha q}$ and the internal randomness of \mathcal{A} . We say that $\mathsf{LWE}_{q,m,n,\alpha}$ is hard if for all PPT algorithm \mathcal{A} , the advantage $\mathbf{Adv}_{q,m,n,\alpha}^{\mathsf{LWE}}(\mathcal{A})$ is negligible in λ .

We require that $\alpha \ge 2\sqrt{n}/q$ for the reduction from worst-case lattice problems and refer the readers to, *e.g.*, [BV16] for more details.

Constructions. For the ease of comparison with [LT19], we now use ℓ to denote the number of clients and n_0 to denote the dimension of a vector \mathbf{x}_i encrypted by client $i \in [\ell]$. For $\lambda \in \mathbb{N}$, let $\mathcal{R}_{\lambda} = \mathbb{Z}$, $\mathcal{K}_{\lambda,\text{pub}} = \{(y, \ldots, y) : y \in [-B; B]\} \subsetneq [-B; B]^{n_0}$, $\mathcal{M}_{\lambda,\text{pri}} = [-B; B]^{n_0}$ and $\mathcal{K}_{\lambda,\text{pri}} = \mathcal{M}_{\lambda,\text{pub}} = \{\top\}$ for polynomials $B = B(\lambda)$ and $n_0 = n_0(\lambda) : \mathbb{N} \to \mathbb{N}$. The functionality $f^{\text{ip}} = \{f_{\lambda,n}^{\text{ip}}\}_{\lambda,n\in\mathbb{N}}$ is defined via

$$f_{\lambda,\ell}^{\mathsf{ip}}\big(\{k_i = (\top, \mathbf{y}_i)\}_{i \in [\ell]}, \{m_i = (\mathbf{x}_i, \top)\}_{i \in [\ell]}\big) = \sum_{i \in [\ell]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle$$

for all $\lambda, \ell \in \mathbb{N}$. Note that this functionality is more restrictive than Definition 12, which considers $\mathcal{K}_{\lambda,\text{pub}} = [-B; B]$. The construction of [LT19] does not quite satisfy the definition of dynamizability, however we can achieve this property with a simple modification. Below we recall their scheme $[\mathsf{FE}]$ and present our modified scheme $[\mathsf{FE}]$.

Construction 57 (DMCFE Scheme [FE] of [LT19] and our Variant [FE]). The constructions are defined w.r.t. the following common global parameters

$$\mathsf{CP} = (\ell_{\max}, n_0, n_1, \overline{n}_1, n, \overline{n}, m, \overline{m}, \alpha, \alpha_1, \overline{\alpha}_1, \sigma, \overline{\sigma}, \ell_t, \ell_f, L, q, \overline{q}, \mathsf{AHF}, \mathsf{AHF}_f, [\underline{M}])$$

where [M] is a new additional parameter for our variant $[\overline{FE}]$. The gadget matrices are defined as:

$$\overline{\mathbf{G}} = [\mathbf{I}_n \otimes (1, 2, 4, \dots, 2^{\lceil \log \overline{q} \rceil}) \mid \mathbf{0}^n \mid \dots \mid \mathbf{0}^n] \in \mathbb{Z}_{\overline{q}}^{n \times \overline{m}}$$

$$\mathbf{G}_0 = [\mathbf{I}_{n_0} \otimes (1, 2, 4, \dots, 2^{\lceil \log \overline{q} \rceil}) \mid \mathbf{0}^{n_0} \mid \dots \mid \mathbf{0}^{n_0}] \in \mathbb{Z}_q^{n_0 \times m}$$

$$\mathbf{G} = [\mathbf{I}_n \otimes (1, 2, 4, \dots, 2^{\lceil \log \overline{q} \rceil}) \mid \mathbf{0}^n \mid \dots \mid \mathbf{0}^n] \in \mathbb{Z}_q^{n \times m} .$$

The constructions $\begin{bmatrix} \mathsf{FE} \end{bmatrix}$ and $\begin{bmatrix} \mathsf{FE} \end{bmatrix}$ work as follows:

Setup $(1^{\lambda}, 1^{\ell}, CP)$: On input the common global parameters CP and the number of clients 1^{ℓ} , sample random matrices:

$$\mathbf{V} \stackrel{s}{\leftarrow} \mathbb{Z}_{q}^{n_{0} \times n} \qquad \left\{ \mathbf{A}_{i,b} \stackrel{s}{\leftarrow} \mathbb{Z}_{q}^{n \times m} \right\}_{i \in [L], b \in \{0,1\}} \\ \overline{\mathbf{V}} \stackrel{s}{\leftarrow} \mathbb{Z}_{q}^{n \times \overline{n}} \qquad \left\{ \mathbf{B}_{i,b} \stackrel{s}{\leftarrow} \mathbb{Z}_{\overline{q}}^{\overline{n} \times \overline{m}} \right\}_{i \in [L], b \in \{0,1\}} ,$$

as well as Gaussian samples $\mathbf{t}_i \stackrel{\text{\tiny{\$}}}{\leftarrow} D_{\mathbb{Z}^{\overline{n}},\overline{\sigma}}$ and $\mathbf{s}_i \stackrel{\text{\tiny{\$}}}{\leftarrow} D_{\mathbb{Z}^n,\sigma}$ for each $i \in [\ell]$. For each $i \in [\ell]$, sample $\mathbf{v}_i \stackrel{\text{\tiny{\$}}}{\leftarrow} [-M, M]^{\overline{n}}$ such that $\sum_{i \in [\ell]} \mathbf{v}_i = 0$. Output $\mathsf{PP} \coloneqq (\mathsf{CP}, \mathbf{V}, \overline{\mathbf{V}}, \{\mathbf{A}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \mathbf{t} \coloneqq \sum_{i=1}^n \mathbf{t}_i)$

$$\left\{\mathsf{SK}_i \coloneqq (\mathbf{s}_i, \mathbf{t}_i, \left[\mathbf{u}_i \coloneqq \mathbf{t}_i + \mathbf{v}_i\right])\right\}_{i \in [\ell]}$$
 .

 $\begin{aligned} \mathsf{KeyGen}(\mathsf{SK}_i, \mathsf{tag-f}, k_i) \colon & \textit{On input a secret key } \mathsf{SK}_i \coloneqq (\mathbf{s}_i, \mathbf{t}_i, [\mathbf{u}_i]), \textit{ a tag } \mathsf{tag-f} \textit{ and } k_i = (\top, y_i) \textit{ compute the hash } \tau_{\mathsf{tag-f}} = \tau_{\mathsf{tag-f}}[1] \dots \tau_{\mathsf{tag-f}}[L] \coloneqq \mathsf{AHF}_{\mathsf{f}}(\mathsf{tag-f}) \in \{0, 1\}^L \textit{ as well as the GSW evaluation} \end{aligned}$

$$\mathbf{B}(\tau_{\mathsf{tag-f}}) = \mathbf{B}_{L,\tau_{\mathsf{tag-f}}[L]} \cdot \overline{\mathbf{G}}^{-1} \Big(\mathbf{B}_{L-1,\tau_{\mathsf{tag-f}}[L-1]} \cdot \overline{\mathbf{G}}^{-1} \big(\cdots \\ \mathbf{B}_{2,\tau_{\mathsf{tag-f}}[2]} \cdot \overline{\mathbf{G}}^{-1} \big(\mathbf{B}_{1,\tau_{\mathsf{tag-f}}[1]} \big) \Big) \Big) \cdot \overline{\mathbf{G}}^{-1} \big(\overline{\mathbf{W}}^{\top} \big) \in \mathbb{Z}_{\overline{q}}^{\overline{n} \times \overline{m}}$$
(5)

and $\overline{\mathbf{W}} = \overline{\mathbf{G}}^{\top} \cdot \overline{\mathbf{V}} \in \mathbb{Z}_{\overline{q}}^{\overline{m} \times \overline{n}}$. Sample $\mathbf{e}_{\mathsf{tag-f},i} \leftarrow D_{\overline{m},\alpha\overline{q}}$ and output

$$\mathsf{DK}_{i} = \left(\mathbf{d}_{i} \coloneqq \overline{\mathbf{G}}^{\top} \cdot (y_{i} \cdot \mathbf{s}_{i}) + \mathbf{B}(\tau_{\mathsf{tag-f}})^{\top} \cdot \mathbf{t}_{i} + \mathbf{e}_{\mathsf{tag-f},i} \in \mathbb{Z}_{\overline{q}}^{\overline{m}}, \left[\bar{\mathbf{u}}_{i}\right]\right)$$

 $\mathsf{Enc}(\mathsf{SK}_i, \mathsf{tag}, \mathbf{x}_i): \text{ On input a secret key } \mathsf{SK}_i \coloneqq (\mathbf{s}_i, \mathbf{t}_i, [\mathbf{u}_i]), \text{ a tag tag and } m_i = (\top, \mathbf{x}_i), \text{ compute the hash } \tau_{\mathsf{tag}} = \tau_{\mathsf{tag}}[1] \dots \tau_{\mathsf{tag}}[L] \coloneqq \mathsf{AHF}(\mathsf{tag}) \in \{0, 1\}^L \text{ as well as the GSW evaluation}$

$$\mathbf{A}(\tau_{\mathsf{tag}}) = \mathbf{A}_{L,\tau_{\mathsf{tag}}[L]} \cdot \mathbf{G}^{-1} \Big(\mathbf{A}_{L-1,\tau_{\mathsf{tag}}[L-1]} \cdot \mathbf{G}^{-1} \big(\cdots \\ \mathbf{A}_{2,\tau_{\mathsf{tag}}[2]} \cdot \mathbf{G}^{-1} \big(\mathbf{A}_{1,\tau_{\mathsf{tag}}[1]} \big) \Big) \Big) \cdot \mathbf{G}^{-1} \big(\mathbf{W}^{\top} \big) \in \mathbb{Z}_q^{n \times m}$$
(6)

and $\mathbf{W} = \mathbf{G}_0^\top \cdot \mathbf{V} \in \mathbb{Z}_q^{m \times n}$. Sample $\mathbf{e}_{\mathsf{tag},i} \stackrel{\mathfrak{s}}{\leftarrow} D_{\mathbb{Z}^m, \alpha q}$ and output

$$\mathsf{CT}_i = \mathbf{G}_0^\top \cdot \mathbf{x}_i + \mathbf{A}(\tau_{\mathsf{tag}})^\top \cdot \mathbf{s}_i + \mathbf{e}_{\mathsf{tag},i}$$
 .

 $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in[\ell]},\{\mathsf{CT}_i\}_{i\in[\ell]}): \text{ On input a set of decryption keys } \{\mathsf{DK}_i = (\mathbf{d}_i, [\mathbf{u}_i])\}_{i\in[\ell]} \text{ and a set of ciphertexts } \{\mathsf{CT}_i\}_{i\in[\ell]}, \text{ compute } [\mathbf{t} = \sum_{i\in[\ell]} \mathbf{u}_i \text{ and }]$

$$\begin{aligned} \tau_{\mathsf{tag-f}} &= \tau_{\mathsf{tag-f}}[1] \dots \tau_{\mathsf{tag-f}}[L] \coloneqq \mathsf{AHF}_{\mathsf{f}}(\mathsf{tag-f}) \in \{0,1\}^{L} \\ \widetilde{\mathbf{d}}_{\mathsf{tag-f}} &= \sum_{i \in [\ell]} \mathbf{d}_{i} - \mathbf{B}(\tau_{\mathsf{tag-f}})^{\top} \cdot \mathbf{t} \mod \overline{q} \end{aligned}$$

where $\mathbf{B}(\tau_{\mathsf{tag-f}})$ is as per (5). Interpret $\mathbf{\widetilde{d}}_{\mathsf{tag-f}} = \mathbf{\overline{G}}^{\top} \mathbf{d}_{\mathsf{tag-f}} + \mathbf{\widetilde{e}}_{\mathsf{tag-f}}$, use the public trapdoor of $\Lambda^{\perp}(\mathbf{\overline{G}})$ to compute $\mathbf{d}_{\mathsf{tag-f}}$. Next, compute

$$\begin{aligned} \tau_{\mathsf{tag}} &= \tau_{\mathsf{tag}}[1] \dots \tau_{\mathsf{tag}}[L] \coloneqq \mathsf{AHF}(\mathsf{tag}) \in \{0,1\}^L \\ \mathbf{z}_{\mathsf{tag}} &= \sum_{i \in [\ell]} y_i \cdot \mathsf{CT}_i - \mathbf{A}(\tau_{\mathsf{tag}})^\top \cdot \mathbf{d}_{\mathsf{tag-f}} \mod q, \end{aligned}$$

where $\mathbf{A}(\tau_{\mathsf{tag}})$ is computed as per (6). Interpret $\mathbf{z}_{\mathsf{tag}} = \mathbf{G}_0^\top \mathbf{z} + \mathbf{e}$, use the public trapdoor of $\Lambda^{\perp}(\mathbf{G}_0)$ to compute $\mathbf{z} \in [-\ell XY, \ell XY]$.

Dynamizability. Intuitively, the original scheme FE is not dynamizable because the vectors $\{\mathbf{t}_i\}_{i\in[n]}$ are conditioned on a global constraint $\sum_{i\in[n]} \mathbf{t}_i = \mathbf{t}$, thus they cannot be sampled by running SetupUser independently for each *i*. To circumvent this problem, we remove \mathbf{t} from the global parameters, include a masked version \mathbf{u}_i of \mathbf{t}_i into each decryption key DK_i and reconstruct \mathbf{t} from $\{\mathbf{u}_i\}_{i\in[n]}$ at decryption time. The following lemma shows that our DMCFE scheme [FE] obtained in this way fits into our framework of dynamizability.

Lemma 58. The DMCFE scheme $[\overline{\mathsf{FE}}]$ in Construction 57 is \mathbb{M} -dynamizable, where $\mathbb{M} = \mathbb{M}(M, \overline{n})$ denotes the finite Abelian group $[-\overline{M}, \overline{M}]^{\overline{n}}$ equipped with modular addition.

Proof. Let $(\mathbf{v}_i)_{i \in [n]} \in \mathcal{S}(n, \mathbb{M})$. The scheme admits the following implementation of the algorithms SetupPP and SetupUser.

Setup $\mathsf{PP}(1^{\lambda})$: Sample random matrices

$$\begin{split} \mathbf{V} &\Leftarrow \mathbb{Z}_q^{n_0 \times n} & \left\{ \mathbf{A}_{i,b} &\Leftarrow \mathbb{Z}_q^{n \times m} \right\}_{i \in [L], b \in \{0,1\}} \\ \overline{\mathbf{V}} &\Leftarrow \mathbb{Z}_q^{n \times \overline{n}} & \left\{ \mathbf{B}_{i,b} &\Leftarrow \mathbb{Z}_{\overline{q}}^{\overline{n} \times \overline{m}} \right\}_{i \in [L], b \in \{0,1\}} , \end{split}$$

then output $\mathsf{PP} \coloneqq (\mathsf{CP}, \mathbf{V}, \overline{\mathbf{V}}, \{\mathbf{A}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}}).$ SetupUser($\mathsf{PP}, \mathbf{v}_i$): Sample $\mathbf{t}_i \stackrel{\text{s}}{\leftarrow} D_{\mathbb{Z}^{\overline{n}}, \overline{\sigma}}$ and $\mathbf{s}_i \stackrel{\text{s}}{\leftarrow} D_{\mathbb{Z}^n, \sigma}$, then output $\mathsf{SK}_i \coloneqq (\mathbf{s}_i, \mathbf{t}_i, \mathbf{u}_i \coloneqq \mathbf{t}_i + \mathbf{v}_i).$

By construction, it is straightforward that for any $\lambda, \ell \in \mathbb{N}$, the distributions

$$\left\{ \mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]} \middle| \begin{array}{c} (\mathbf{v}_i)_{i \in [n]} \xleftarrow{\hspace{0.1cm} \$} \mathcal{S}(n, \mathbb{M}) \\ \mathsf{PP} \leftarrow \mathsf{Setup}\mathsf{PP}(1^{\lambda}) \\ \forall i \in [n] \colon \mathsf{SK}_i \leftarrow \mathsf{SetupUser}(\mathsf{PP}, \mathbf{v}_i) \end{array} \right\}$$

and $\{(\mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\ell)\}$ are identical.

Security. We first recall the security result from [LT19]

Proposition 59 ([LT19]). The DMCFE scheme **FE** presented in Construction 57 is stat-adap-symsecure against complete queries without repetitions under the LWE assumption in the standard model with respect to the following choice of parameters:

- Let $\ell_{\max} = \lambda^k$, $n_1 = \lambda^d$, $\overline{d} = 3d + k 1$, $q = 2^{\lambda^{d-1} + \lambda}$, $q = 2^{\lambda^{\overline{d} 1} + \lambda}$, $\overline{n}_1 = \lambda^{\overline{d}}$, $\alpha_1 = 2^{-\lambda^{d-1} + d\log\lambda}$, $\overline{\alpha}_1 = 2^{-\lambda^{\overline{d} 1} + \overline{d}\log\lambda}$, $\alpha = 2^{-\sqrt{\lambda}}$, $n_0 \cdot \ell_{\max} = O(\lambda^{d-2})$, $n_0 = O(\lambda^{d-2})$, $n = O(\lambda^{2d-1})$, $\overline{n} = O(\lambda^{4d+k-2})$, $\sigma = 2^{\lambda^{\overline{d} 1} 2\lambda}$, $\overline{\sigma} = 2^{\lambda^{\overline{d} 1} 2\lambda}$, and $m, \overline{m} = \operatorname{poly}(\lambda)$.
- The tag lengths $\ell_t \in \Theta(\lambda)$ for encryption and $\ell_f \in \Theta(\lambda)$ for key generation.
- The dimensions $n, m, n_0, n_1, \overline{n}, \overline{m} \in \text{poly}(\lambda)$ satisfy that $n > 3 \cdot (n_0 + n_1) \cdot \lceil \log q \rceil, m > 2 \cdot n \cdot \lceil \log q \rceil, \overline{n} > 3 \cdot (n + n_1) \cdot \lceil \log \overline{q} \rceil$, and $\overline{m} > 2 \cdot \overline{n} \cdot \lceil \log \overline{q} \rceil$.
- The description of balanced admissible hash functions $\mathsf{AHF} : \{0,1\}^{\ell_t} \to \{0,1\}^L$ and $\mathsf{AHF}_f : \{0,1\}^{\ell_f} \to \{0,1\}^L$ for suitable $L \in \mathrm{poly}(\lambda)$.
- A real $\alpha > 0$ and a Gaussian parameter $\sigma > 0$ so that the interval $[-\beta, \beta] \coloneqq [-\sigma \sqrt{n}, \sigma \sqrt{n}]$ specifies the domain for the secret vector's coordinates (with overwhelming probability).
- The real $[M] \coloneqq \lambda^2 \cdot \overline{\sigma}$.

Based on this result, we prove security of our variant $[\overline{\mathsf{FE}}]$. Intuitively, the only relevant difference between $[\overline{\mathsf{FE}}]$ and $[\overline{\mathsf{FE}}]$ is that, for $i \in [n]$, the decryption key DK_i in $[\overline{\mathsf{FE}}]$ additionally contains a masked version \mathbf{u}_i of \mathbf{t}_i . As the vector \mathbf{t}_i is part of the secret key SK_i in the original scheme $[\overline{\mathsf{FE}}]$, we must show that \mathbf{u}_i does not leak any information about \mathbf{t}_i . If \mathbf{t}_i was sampled from a finite set S, the argument would be trivial: we could simply pick the mask \mathbf{v}_i according to the uniform distribution over S. However, the distribution of \mathbf{t}_i follows a discrete Gaussian over the infinite set $\mathbb{Z}^{\overline{n}}$ making the argument slightly more complex. First, we use the Chernoff-Cramér tail bound to argue that \mathbf{t} is in a bounded-size interval with high probability. Second, we show that when choosing the mask \mathbf{v}_i from this interval, then the vector \mathbf{u}_i statistically hides \mathbf{t}_i .

Proposition 60. If the DMCFE scheme [FE] in 57 is stat-adap-sym-secure against complete queries without repetitions under the LWE assumption in the standard model, then so is our modified scheme [FE].

Proof. Let \mathcal{A} be an adversary in the security game $\mathbf{Exp}_{[\bar{\mathsf{FE}}], f^{\mathsf{ip}}, \mathcal{A}}^{\mathsf{dmcfe}-b}(1^{\lambda})$ attacking the security of $[\bar{\mathsf{FE}}]^{\cdot}$. We build an adversary \mathcal{B} in the security game $\mathbf{Exp}_{[\bar{\mathsf{FE}}], f^{\mathsf{ip}}, \mathcal{B}}^{\mathsf{dmcfe}-b}(1^{\lambda})$ that uses \mathcal{A} as a subroutine and attacks the security of the original scheme $[\bar{\mathsf{FE}}]$. For convenience, we introduce the shorthands

$$\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}} \coloneqq \mathbf{Exp}_{\mathcal{B}}^{\mathsf{dmcfe-}b} = \mathbf{Exp}_{\mathcal{B}}^{\mathsf{dm$$

The adversary \mathcal{B} works as follows:

• Initialization and Static Corruption Queries: Upon \mathcal{A} calling $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}}$.Initialize (1^{λ}) , \mathcal{B} calls the initialization procedure $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{ip}}$.Initialize (1^{λ}) of its own challenger to obtain

$$\mathsf{PP} \coloneqq \left(\mathsf{CP}, \mathbf{V}, \overline{\mathbf{V}}, \{\mathbf{A}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}}, [\mathbf{t}]\right) \ .$$

and sends $\left[\bar{\mathbf{P}} \bar{\mathbf{P}} \right]^{-} = (\mathsf{CP}, \mathbf{V}, \overline{\mathbf{V}}, \{\mathbf{A}_{i,b}\}_{i \in [L], b \in \{0,1\}}, \{\mathbf{B}_{i,b}\}_{i \in [L], b \in \{0,1\}})$ to \mathcal{A} . The adversary \mathcal{B} aborts if $\mathbf{t} \notin [-M, \bar{M}]^{\overline{n}}$.

In the static corruption setting, \mathcal{A} declares up front a set $\mathcal{C} \subset [\ell]$ of corrupted clients. For each $i \in \mathcal{C}$, \mathcal{B} samples $\mathbf{v}_i \notin [-M, M]^{\overline{n}}$, queries $(\mathbf{s}_i, \mathbf{t}_i) \leftarrow \mathbf{Exp}_{\mathcal{B}}^{\mathrm{ip}}.\mathcal{O}\mathsf{Corrupt}(i)$ and returns $\mathsf{SK}_i = (\mathbf{s}_i, \mathbf{t}_i, [\mathbf{u}_i \coloneqq \mathbf{t}_i + \mathbf{v}_i])$. Finally, for each $i \in \mathcal{H} \coloneqq [\ell] \setminus \mathcal{C}$, \mathcal{B} samples vectors $\mathbf{u}_i \notin [-M, M]^{\overline{n}}$ conditioned on $\sum_{i \in \mathcal{H}} \mathbf{u}_i = \mathbf{t} - \sum_{i \in \mathcal{C}} \mathbf{u}_i$.

• Encryption Queries: Upon receiving a query

$$\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}}.\mathcal{O}\mathsf{Enc}(i,\mathsf{tag},m_i^{(0)}=(\top,\mathbf{x}_i^{(0)}),m_i^{(1)}=(\top,\mathbf{x}_i^{(1)})) \ ,$$

if $i \in \mathcal{H}$, \mathcal{B} returns $\mathsf{CT}_i \leftarrow [FE].\mathsf{Enc}((\mathbf{s}_i, \mathbf{t}_i), \mathsf{tag}, m_i^{(b)})$ computed by running the encryption algorithm of [FE]. If $i \in \mathcal{C}$, \mathcal{B} queries $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{ip}}.\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, \mathbf{m}_i^{(0)}, \mathbf{m}_i^{(1)})$ and forwards the result to \mathcal{A} .

• Key-Generation Queries: Upon receiving a query

$$\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}.\mathcal{O}\mathsf{KeyGen}(i,\mathsf{tag-f},k^{(0)}_i=(\mathbf{y}^{(0)}_i,\top),k^{(1)}_i=(\mathbf{y}^{(1)}_i,\top)) \ ,$$

if $i \in \mathcal{H}$, \mathcal{B} returns $\mathsf{DK}_i \leftarrow \overline{\mathsf{FE}}$. KeyGen $((\mathbf{s}_i, \mathbf{t}_i), \mathsf{tag-f}, k_i^{(b)})$ computed by running the encryption algorithm of $\overline{\mathsf{FE}}$. If $i \in \mathcal{C}$, \mathcal{B} calls $\mathbf{d}_i \leftarrow \mathbf{Exp}_{\mathcal{B}}^{\mathsf{ip}}.\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag-f}, k_i^{(0)}, k_i^{(1)})$ and sends $\left[\underbrace{\mathsf{DK}_i}_{\mathsf{L}} \right] = (\mathbf{d}_i, \mathbf{u}_i)$ to \mathcal{A} .

• Finalize: Upon \mathcal{A} calling $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}}$.Finalize(b'), \mathcal{B} calls $\mathbf{Exp}_{\mathcal{B}}^{\mathsf{ip}}$.Finalize(b').

For $\mathbf{t} = \sum_{i \in [\ell]} \mathbf{t}_i$ with $\mathbf{t}_i \stackrel{\text{s}}{\leftarrow} D_{\mathbb{Z}^{\overline{n}},\overline{\sigma}}$ *i.i.d*, it holds that \mathbf{t} follows the Gaussian distribution $D_{\mathbb{Z}^{\overline{n}},\ell\cdot\overline{\sigma}}$ where the standard deviation is multiplied by a factor ℓ . By using the fact that the center of $D_{\mathbb{Z}^{\overline{n}},n\cdot\overline{\sigma}}$ is **0** together with the union bound, the Chernoff-Cramér bound in Equation (3) yields:

$$\Pr[\mathcal{B} \text{ aborts on } \mathbf{t}] = \Pr\left[\exists i \in [\ell] : |\mathbf{t}[i]| \ge M\right] \le \ell \cdot \left(2\exp\left(-\frac{M^2}{2\ell^2 \overline{\sigma}^2}\right)\right) ,$$

which is negligible in λ under the parameter choice $M = \lambda^2 \cdot \overline{\sigma}$ with respect to ℓ and $\overline{\sigma}$. In what follows, we condition on the event that \mathcal{B} does not abort on \mathbf{t} . By construction the public parameters, encryption responses, and the *corrupted* keys provided by \mathcal{B} are identical to those in the experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}}$. It therefore suffices to show that the $\mathcal{O}\mathsf{D}\mathsf{KeyGen}$ responses $\left[\overset{\frown}{\mathsf{D}}\overset{\frown}{\mathsf{K}}_{i} \right] = (\mathbf{d}_{i}, \mathbf{u}_{i})$ simulated by \mathcal{B} are indistinguishable from those in the experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}}$. Specifically, we show that the following distributions are statistically close

$$D_{0} \coloneqq \left\{ \begin{aligned} \left\{ (\mathbf{t}_{i}, \mathbf{v}_{i}) \right\}_{i \in \mathcal{C}}; \\ \left\{ \mathbf{u}_{i} \right\}_{i \in \mathcal{H}}; \mathbf{t} \end{aligned} \middle| \begin{array}{l} \forall i \in [\ell] : \mathbf{t}_{i} \stackrel{\$}{\leftarrow} D_{\mathbb{Z}^{\overline{n}}, \overline{\sigma}}; \ \mathbf{t} \coloneqq \sum_{i \in [\ell]} \mathbf{t}_{i} \\ \forall i \in \mathcal{C} : \mathbf{v}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in \mathcal{H} : \mathbf{u}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in \mathcal{H} : \mathbf{u}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in \mathcal{H} : \mathbf{u}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in [\ell] : \mathbf{t}_{i} \stackrel{\$}{\leftarrow} D_{\mathbb{Z}^{\overline{n}}, \overline{\sigma}}; \ \mathbf{t} \coloneqq \sum_{i \in [\ell]} \mathbf{t}_{i} \\ \hline \forall i \in [\ell] : \mathbf{t}_{i} \stackrel{\$}{\leftarrow} D_{\mathbb{Z}^{\overline{n}}, \overline{\sigma}}; \ \mathbf{t} \coloneqq \sum_{i \in [\ell]} \mathbf{t}_{i} \\ \hline \forall i \in [\ell] : \mathbf{v}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in [\ell] : \mathbf{v}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in [\ell] : \mathbf{v}_{i} \stackrel{\$}{\leftarrow} [-M, M]^{\overline{n}} \\ \hline \forall i \in \mathcal{H} : \mathbf{u}_{i} \coloneqq \mathbf{t}_{i} + \mathbf{v}_{i} \end{aligned} \right\},$$

where D_0 corresponds to the simulation of \mathcal{B} and D_1 corresponds to the responses in the experiment $\mathbf{Exp}_{\mathcal{A}}^{\mathsf{ip}}$.

We recall that U(S) denotes the uniform distribution on a finite set S, and that $\mathcal{S}(\ell, [-M, M]^{\overline{n}})$ denotes the distribution that outputs $\mathbf{v}_i \stackrel{*}{\leftarrow} [-M, M]^{\overline{n}}$ for $i \in [\ell]$ conditioned on $\sum_{i \in [\ell]} \mathbf{v}_i = 0$. W.l.o.g, we extend the distribution $U([-M, M]^{\overline{n}})$ over $[-M, M]^{\overline{n}}$ to a distribution over $\mathbb{Z}^{\overline{n}}$ such that for any $\mathbf{x} \in \mathbb{Z}^{\overline{n}} \setminus [-M, M]^{\overline{n}}$ it holds $\Pr_{X \sim U([-M,M]^{\overline{n}})}[X = \mathbf{x}] = 0$. For the ease of notation, we use boldface letters \mathbf{T} , \mathbf{U} and \mathbf{V} to denote collections of vectors $\{\mathbf{t}_i\}_i, \{\mathbf{u}_i\}_i, \{\mathbf{v}_i\}_i$ sampled following a given distribution.

We will make use of the following lemma which is proven below.

Lemma 61. For each $\mathbf{U} \in ([-M, M]^{\overline{n}})^{|\mathcal{H}|}$, we have

$$\Pr[D_1 \to \mathbf{U}] \le \frac{(2M+1)^{\overline{n}}}{(2M+1)^{\overline{n}|\mathcal{H}|}} \left(1 + \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^2}{2\overline{\sigma}^2 - |\mathcal{H}|\pi\overline{n}\overline{B}^2}\right) + \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1}$$

where $\overline{B} \coloneqq \lambda \cdot \overline{\sigma}$.

We bound the statistical distance $\Delta(D_0, D_1)$ as follows:

$$\begin{aligned}
\Delta(D_0, D_1) & (7) \\
\stackrel{(\star)}{\leq} & \max_{\substack{\mathcal{S} \subseteq ([-M,M]^{\overline{n}})^{|\mathcal{H}|} \\ \forall \mathbf{U} \in \mathcal{S} : \Pr[D_1 \to \mathbf{U}] > \Pr[D_0 \to \mathbf{U}]}} \left| \Pr[D_1 \in \mathcal{S}] - \Pr[D_0 \in \mathcal{S}] \right| + \operatorname{negl}_1(\lambda) \\
\begin{pmatrix} (\star) \\ \leq \\ & \sum_{\substack{\mathbf{U} \in \mathcal{S} \subseteq ([-M,M]^{\overline{n}})^{|\mathcal{H}|} \\ \forall \mathbf{U} \in \mathcal{S} : \Pr[D_1 \to \mathbf{U}] > \Pr[D_0 \to \mathbf{U}]}} \left(\Pr[D_1 \to \mathbf{U}] - \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} \right) + \operatorname{negl}_1(\lambda) \\
\end{aligned}$$
(7)
$$(7)$$

We note that (\star) follows from the definition of the statistical distance and Equation (3), and (\blacktriangle) applies the uniform choice of U in D_0 . Next, we use Lemma 61 and obtain from (8) that

$$\begin{split} \Delta(D_{0}, D_{1}) \\ &\leq \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\overline{n}})^{|\mathcal{H}|}\\\forall\mathbf{U}\in\mathcal{S}:\Pr[D_{1}\rightarrow\mathbf{U}]>\Pr[D_{0}\rightarrow\mathbf{U}]}} \left(\frac{(2M+1)^{\overline{n}}}{(2M+1)^{\overline{n}|\mathcal{H}|}} \cdot \left(1 + \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^{2}}{2\overline{\sigma}^{2} - |\mathcal{H}|\pi\overline{n}\overline{B}^{2}}\right) \\ &\quad + \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} - \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1}\right) + \operatorname{negl}_{1}(\lambda) \\ &\leq \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\overline{n}})^{|\mathcal{H}|}\\\forall\mathbf{U}\in\mathcal{S}:\Pr[D_{1}\rightarrow\mathbf{U}]>\Pr[D_{0}\rightarrow\mathbf{U}]}} \frac{(2M+1)^{\overline{n}}}{(2M+1)^{\overline{n}|\mathcal{H}|}} \cdot \left(1 + \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^{2}}{2\overline{\sigma}^{2} - |\mathcal{H}|\pi\overline{n}\overline{B}^{2}}\right) + \operatorname{negl}_{1}(\lambda) \\ &\leq \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\overline{n}})^{|\mathcal{H}|}\\\forall\mathbf{U}\in\mathcal{S}:\Pr[D_{1}\rightarrow\mathbf{U}]>\Pr[D_{0}\rightarrow\mathbf{U}]}} \frac{1}{(2M+1)^{\overline{n}|\mathcal{H}|}} \cdot \exp\left(2M\overline{n} - \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^{2}}{|\mathcal{H}|\pi\overline{n}\overline{B}^{2} - 2\overline{\sigma}^{2}}\right) \\ &\quad + \operatorname{negl}_{1}(\lambda) \\ &\leq \exp\left(2M\overline{n} - \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^{2}}{|\mathcal{H}|\pi\overline{n}\overline{B}^{2} - 2\overline{\sigma}^{2}}\right) + \operatorname{negl}_{1}(\lambda) , \end{split}$$

where (3) uses the fact that $1 + x \leq e^x$ from (4). To bound $\operatorname{negl}_1(\lambda)$, we can perform a similar calculation as it is done for $\operatorname{negl}_2(\lambda)$ in the proof of Lemma 61. By parameter choices $|\mathcal{H}|\pi\overline{n}\overline{B}^2 = \omega(1)\overline{\sigma}^2$ and $M = o(|\mathcal{H}|\pi\overline{B}^2)$, then $\exp\left(2M\overline{n} - \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^2}{|\mathcal{H}|\pi\overline{n}\overline{B}^2 - 2\overline{\sigma}^2}\right)$ is negligible in λ and the proof is completed.

,

We now prove the lemma.

Proof (of Lemma 61). For $\mathbf{U} \in ([-M, M]^{\overline{n}})^{|\mathcal{H}|}$, noting that in D_1 each $\mathbf{t}_i \stackrel{*}{\leftarrow} D_{\mathbb{Z}^{\overline{n}}, \overline{\sigma}}$ is *i.i.d* by construction, we compute

$$\begin{aligned}
&\operatorname{Pr}[D_{1} \to \mathbf{U}] \\
&\leq \sum_{\substack{\mathbf{T} \in ([-B,B]^{\overline{n}})^{|\mathcal{H}|} \\ \mathbf{V} \in ([-M,M]^{\overline{n}})^{|\mathcal{H}|} \\ \mathbf{T} + \mathbf{V} = \mathbf{U}}} \operatorname{Pr}[\forall i \in \mathcal{H} : D_{\mathbb{Z}^{\overline{n}},\overline{\sigma}} \to \mathbf{T}[i]] \cdot \operatorname{Pr}[\mathcal{S}(n, [-M,M]^{\overline{n}}) \to \mathbf{V}] + \operatorname{negl}_{2}(\lambda) \\
&= \sum_{\substack{\mathbf{T} \in ([-B,B]^{\overline{n}})^{|\mathcal{H}|} \\ \mathbf{V} \in ([-M,M]^{\overline{n}})^{|\mathcal{H}|} \\ \mathbf{T} + \mathbf{V} = \mathbf{U}}} \left(\prod_{i=1}^{|\mathcal{H}|} \frac{\rho_{\overline{\sigma}}(\mathbf{T}[i])}{\rho_{\overline{\sigma}}(\mathbb{Z}^{\overline{n}})} \right) \cdot \left(\frac{1}{(2M+1)^{\overline{n}}} \right)^{|\mathcal{H}|-1} + \operatorname{negl}_{2}(\lambda) \end{aligned} \tag{9}$$

We denote the *i*-th vector in **T** by $\mathbf{T}[i] \in [-B, B]^{\overline{n}}$ and write $\boldsymbol{\Sigma} := \overline{\sigma}^2 \cdot \mathbf{I}$. Then evaluating the Gaussian term gives

$$\frac{\rho_{\overline{\sigma}}(\mathbf{T}[i])}{\rho_{\overline{\sigma}}(\mathbb{Z}^{\overline{n}})} \leq \frac{\rho_{\overline{\sigma}}(\mathbf{T}[i])}{\rho_{\overline{\sigma}}([-B,B]^{\overline{n}})} = \frac{\exp(-\pi\mathbf{T}[i]^{\top}\mathbf{\Sigma}^{-1}\mathbf{T}[i])}{\sum_{\mathbf{T}\in[-B,B]^{\overline{n}}}\exp(-\pi\mathbf{T}^{\top}\mathbf{\Sigma}^{-1}\mathbf{T})} \\
= \frac{\exp(-\pi\|\mathbf{T}[i]\|_{2}^{2}/\overline{\sigma}^{2})}{\sum_{\mathbf{T}\in[-B,B]^{\overline{n}}}\exp(-\pi\|\mathbf{T}\|_{2}^{2}/\overline{\sigma}^{2})} \\
\stackrel{(a)}{\leq} \frac{\exp(\pi\overline{n}\overline{B}^{2}/\overline{\sigma}^{2})}{(2\overline{B}+1)^{\overline{n}}} \\
\stackrel{(b)}{\leq} \frac{\left(1 + \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^{2}}{2\overline{\sigma}^{2} - |\mathcal{H}|\pi\overline{n}\overline{B}^{2}}\right)^{1/|\mathcal{H}|}}{(2\overline{B}+1)^{\overline{n}}}, \qquad (10)$$

where (a) follows from the fact that the squared Euclidean norm $\|\mathbf{T}[i]\|_2^2$ is bounded by \overline{nB}^2 , and (b) uses the inequality $e^x \leq (1 + \frac{x}{n})^{n+x/2}$ from (4). We observe that any choice of \mathbf{T} given \mathbf{U} fixes \mathbf{V} . Then plugging (10) into (9) implies

$$\Pr[D_1 \to \mathbf{U}] \leq \left(\frac{2\overline{B}+1}{2M+1}\right)^{\overline{n} \cdot |\mathcal{H}|} \cdot (2M+1)^{\overline{n}} \left(\frac{1}{(2\overline{B}+1)^{\overline{n}}}\right)^{|\mathcal{H}|} \\ \cdot \left(1 + \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^2}{2\overline{\sigma}^2 - |\mathcal{H}|\pi\overline{n}\overline{B}^2}\right) + \operatorname{negl}_2(\lambda) \\ \leq \frac{(2M+1)^{\overline{n}}}{(2M+1)^{\overline{n}|\mathcal{H}|}} \left(1 + \frac{2|\mathcal{H}|\pi\overline{n}\overline{B}^2}{2\overline{\sigma}^2 - |\mathcal{H}|\pi\overline{n}\overline{B}^2}\right) + \operatorname{negl}_2(\lambda) \quad .$$

It remains to bound $\operatorname{negl}_2(\lambda)$. By employing the independence of the mask V from T, we have

$$\begin{split} \operatorname{negl}_{2}(\lambda) &\leq \sum_{\mathbf{V} \in ([-M,M])^{|\mathcal{H}|}} \operatorname{Pr}[\mathcal{S}(\ell, [-M,M]^{\overline{n}}) \to \mathbf{V}] \cdot \operatorname{Pr}[\mathbf{T} \notin ([-B,B]^{\overline{n}})^{|\mathcal{H}|} \\ &\stackrel{(c)}{\leq} (2M+1)^{\overline{n}|\mathcal{H}|} \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} \cdot \operatorname{Pr}[\mathbf{T} \notin ([-B,B]^{\overline{n}})^{|\mathcal{H}|}] \\ &\stackrel{(d)}{\leq} (2M+1)^{\overline{n}|\mathcal{H}|} \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} \cdot 2\overline{n}|\mathcal{H}| \cdot \exp\left(-\frac{\overline{B}^{2}}{2\overline{\sigma}^{2}}\right) \\ &\stackrel{(e)}{\leq} \exp\left(2M\overline{n}|\mathcal{H}|\right) \cdot \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} \cdot 2\overline{n}|\mathcal{H}| \cdot \exp\left(-\frac{\overline{B}^{2}}{2\overline{\sigma}^{2}}\right) \\ &= \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} \cdot 2\overline{n}|\mathcal{H}| \cdot \exp\left(2M\overline{n}|\mathcal{H}| - \frac{\overline{B}^{2}}{2\overline{\sigma}^{2}}\right) \\ &\stackrel{(f)}{\leq} \left(\frac{1}{(2M+1)^{\overline{n}}}\right)^{|\mathcal{H}|-1} , \end{split}$$

where (c) uses the union bound over all possible values of **V** and the fact that it is uniformly distributed, (d) employs the Gaussian distribution of **T** and Equation (3), (e) uses the inequality $1 + x \leq e^x$ from (4) and (f) follows from the parameter choice $M = o(\overline{B}^2)$. This concludes the proof.

Combining Theorem 46 and Proposition 60 with the generic conversion from DMCFE to DDFE (Construction 25, Proposition 26), we obtain the following corollary.

Corollary 62. Assuming LWE in the standard model, there exists a DDFE scheme for f^{ip} , where functional vectors have the same entry in each coordinate, that is stat-adap-sym-secure without repetitions.

A.5 Instantiation of Construction 25 with the DMCFE of [NPS24]

The construction of [NPS24] is based on the *dual pairing vector spaces* (DPVS) framework which we briefly recall below.

Dual Pairing Vector Spaces. Let $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, q)$ be a pairing group, $N \in \mathbb{N}$ and consider \mathbb{G}_1^N having N copies of \mathbb{G}_1 . Viewing \mathbb{Z}_q^N as a vector space of dimension N over \mathbb{Z}_q with the notions of bases, we can obtain naturally a similar notion of bases for \mathbb{G}_1^N . More specifically, any invertible matrix $B \in GL_N(\mathbb{Z}_q)$ identifies a basis \mathbf{B} of \mathbb{G}_1^N , whose *i*-th row \mathbf{b}_i is $[B_i]_1$, where B_i is the *i*-th row of B. It is straightforward that we can write $\mathbf{B} = [B]_1$ for any basis \mathbf{B} of \mathbb{G}_1^N corresponding to an invertible matrix $B \in GL_N(\mathbb{Z}_q)$. We write $\mathbf{x} = (m_1, \ldots, m_N)_{\mathbf{B}}$ to indicate the representation of \mathbf{x} in the basis \mathbf{B} , i.e. $\mathbf{x} = \sum_{i=1}^N m_i \cdot \mathbf{b}_i$. Treating \mathbb{G}_2^N similarly, we can furthermore define a product of two vectors $\mathbf{x} = [(m_1, \ldots, m_N)]_1 \in \mathbb{G}_1^N, \mathbf{y} = [(k_1, \ldots, k_N)]_2 \in \mathbb{G}_2^N$ by $\mathbf{x} \times \mathbf{y} \coloneqq \prod_{i=1}^N e(\mathbf{x}[i], \mathbf{y}[i]) = [\langle (m_1, \ldots, m_N), (k_1, \ldots, k_N) \rangle]_1$. Given a basis $\mathbf{B} = (\mathbf{b}_i)_{i \in [N]}$ of \mathbb{G}_1^N , we define \mathbf{B}^* to be a basis of \mathbb{G}_2^N by first defining $B^* \coloneqq (B^{-1})^\top$ and the *i*-th row \mathbf{b}_i^* of \mathbf{B}^* is $[B_i^*]_2$. It holds that $B \cdot (B^*)^\top = I_N$ the identity matrix and $\mathbf{b}_i \times \mathbf{b}_j^* = [\delta_{i,j}]_1$ for every $i, j \in [N]$, where $\delta_{i,j} = 1$ if and only if i = j. We call the pair $(\mathbf{B}, \mathbf{B}^*)$ a pair of dual orthogonal bases of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$. If \mathbf{B} is constructed by a random invertible matrix $B \stackrel{\&}{\leftarrow} GL_N(\mathbb{Z}_q)$, we call the resulting $(\mathbf{B}, \mathbf{B}^*)$ a pair of random dual bases. A DPVS is a bilinear group $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, q)$ with dual orthogonal bases. We denote by DPVSGen the algorithm that takes as inputs \mathbb{G} , and a unary 1^N , then outputs a pair of random matrices (B, B^*) that specify dual orthogonal bases $(\mathbf{B} = \llbracket B \rrbracket_1, \mathbf{B}^* = \llbracket B^* \rrbracket_2)$ of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$.

We recall the construction of [NPS24] for the functionality $f^{\text{fh-ip}}$. Assuming SXDH in the ROM, the construction is stat-adap-sym-secure in the one-challenge setting under the complete-queries constraint against unbounded repetitions for \mathcal{O} Enc queries and $J = \text{poly}(\lambda)$ repetitions for \mathcal{O} KeyGen queries. The parameter J must be specified at Setup time.

Construction 63 (DMCFE scheme of [NPS24]). The construction is based on a pairing group $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, e, q)$. The details of the scheme $\mathsf{FE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ go as follows:

Setup $(1^{\lambda}, 1^{n})$: On input the security parameter 1^{λ} and the number of clients 1^{n} , sample two fulldomain hash functions H_{1} and H_{2} onto \mathbb{G}_{1} and \mathbb{G}_{2} , respectively. Furthermore, sample matrices $(B_{i}, B_{i}^{*}) \leftarrow \mathsf{DPVSGen}(\mathbb{G}, 1^{2N \cdot (J+1)+4})$, for $i \in [n]$, that specify dual orthogonal bases $(\mathbf{B}_{i}, \mathbf{B}_{i}^{*})$.¹⁶ Sample $(\tilde{t}_{i})_{i} \stackrel{s}{\leftarrow} \mathbb{Z}_{q}^{n}$ conditioned on $\sum_{i=1}^{n} \tilde{t}_{i} = 0$. Output the public parameters $\mathsf{PP} \coloneqq (\mathsf{H}_{1}, \mathsf{H}_{2})$ and the secret keys $\{\mathsf{SK}_{i}\}_{i \in [n]}$ as follows:

$$\mathsf{SK}_{i} \coloneqq (\tilde{t}_{i}, \mathbf{b}_{i,1}, \dots, \mathbf{b}_{i,N}, B_{i,N+1}, \mathbf{b}_{i,N+3}, \mathbf{b}_{i,1}^{*}, \dots, \mathbf{b}_{i,N}^{*}, B_{i,N+1}^{*}, \mathbf{b}_{i,N+2}^{*})$$

KeyGen(SK_i, tag-f, $k_i = (\mathbf{y}_i, \top)$): Parse SK_i as above, compute $H_2(tag-f) \rightarrow \llbracket \mu \rrbracket_2$ and sample $\pi_i \stackrel{*}{\leftarrow} \mathbb{Z}_q$. Then output

$$\mathsf{DK}_{i} = \sum_{k=1}^{N} \mathbf{y}_{i}[k] \mathbf{b}_{i,k}^{*} + \llbracket \mu \rrbracket_{2} \cdot B_{i,N+1}^{*} + \pi_{i} \mathbf{b}_{i,N+2}^{*}$$
$$= (\mathbf{y}_{i}, \ \mu, \ \pi_{i}, \ 0, \ 0^{N+2N \cdot J+1})_{\mathbf{B}_{i}^{*}} \ .$$

 $\mathsf{Enc}(\mathsf{SK}_i, \mathsf{tag}, m_i = (\mathbf{x}_i, \top))$: Parse SK_i as above, compute $\mathsf{H}_1(\mathsf{tag}) \to \llbracket \omega \rrbracket_1$ and sample a random scalar $\rho_i \stackrel{s}{\leftarrow} \mathbb{Z}_q$. Then output

$$\begin{aligned} \mathsf{CT}_{i} &= \sum_{k=1}^{N} \mathbf{x}_{i}[k] \mathbf{b}_{i,1} + \tilde{t}_{i} \llbracket \omega \rrbracket_{1} \cdot B_{i,N+1} + \rho_{i} \mathbf{b}_{i,N+3} \\ &= (\mathbf{x}_{i}, \ \tilde{t}_{i} \omega, \ 0, \ \rho_{i}, \ 0^{N+2N \cdot J+1})_{\mathbf{B}_{i}} \ . \end{aligned}$$

 $\mathsf{Dec}(\{\mathsf{DK}_i\}_{i\in[n]}, \{\mathsf{CT}_i\}_{i\in[n]}): Compute \, \llbracket d \rrbracket_{\mathsf{t}} = \sum_{i=1}^n \mathsf{CT}_i \times \mathsf{DK}_i, \text{ then find and output the discrete log} d.$

Proposition 64 ([NPS24]). Construction 63 is one-challenge stat-adap-sym-secure against complete queries with unbounded repetitions for $\mathcal{O}Enc$ queries and polynomially bounded repetitions for $\mathcal{O}KeyGen$ queries under the SXDH assumption in the ROM.

The following lemma argues that the construction fits into the framework of dynamizable DMCFE schemes.

Lemma 65. The DMCFE scheme FE in Construction 63 is \mathbb{Z}_q -dynamizable.

Proof. Let $(\tilde{t}_i)_{i \in [n]} \in \mathcal{S}(n, \mathbb{Z}_q)$. The scheme admits the following implementation of the algorithms SetupPP and SetupUser.

¹⁶ For each $i \in [n]$, we denote *j*-th row of \mathbf{B}_i (resp. \mathbf{B}_i^*) by $\mathbf{b}_{i,j}$ (resp. $\mathbf{b}_{i,j}^*$). Similarly, $B_{i,k}$ (respectively $B_{i,k}^*$) denotes the *k*-th row of the basis changing matrix B_i (respectively B_i^*).

SetupPP(1^{λ}): Sample two full-domain hash functions H₁ and H₂ onto \mathbb{G}_1 and \mathbb{G}_2 respectively. Return PP := (H₁, H₂).

SetupUser(PP, \tilde{t}_i): Generate $(B_i, B_i^*) \leftarrow DPVSGen(\mathbb{G}, 1^{4N+5})$ and return SK_i computed as follows:

$$\mathsf{SK}_i \coloneqq (\tilde{t}_i, \mathbf{b}_{i,1}, \dots, \mathbf{b}_{i,N}, \ B_{i,N+1}, \ \mathbf{b}_{i,N+3}, \mathbf{b}^*_{i,1}, \dots, \mathbf{b}^*_{i,N}, \ B^*_{i,N+1}, \ \mathbf{b}^*_{i,N+2}) \ .$$

Then the distributions

$$\left\{ \mathsf{PP}, \{\mathsf{SK}_i\}_{i \in [n]} \middle| \begin{array}{c} (\tilde{t}_i)_{i \in [n]} \xleftarrow{\hspace{0.1cm} \$} \mathcal{S}(n, \mathbb{Z}_q) \\ \mathsf{PP} \leftarrow \mathsf{Setup}\mathsf{PP}(1^{\lambda}) \\ \forall i \in [n] \colon \mathsf{SK}_i \leftarrow \mathsf{SetupUser}(\mathsf{PP}, \tilde{t}_i) \end{array} \right\}$$

and $\{(\mathsf{PP}, \{\mathsf{SK}_i\}_{i\in[n]}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^n)\}$ are identical.

Combining Theorem 46 and Proposition 64 with the generic conversion from DMCFE to DDFE (Construction 25, Proposition 26), we obtain the following corollary.

Corollary 66. Assuming SXDH and the ROM, there exists a DDFE scheme for $f^{\text{fh-ip}}$ that is stat-adap-sym-secure with unbounded repetitions for $\mathcal{O}Enc$ queries and polynomially bounded repetitions for $\mathcal{O}KeyGen$ queries.