




# Optimal Security Notion for Decentralized Multi-Client Functional Encryption

Ky Nguyen<sup></sup>, Duong Hieu Phan<sup></sup>, and David Pointcheval<sup></sup>

<sup>1</sup> DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France

<sup>2</sup> LTCI, Telecom Paris, Institut Polytechnique de Paris, France

**Abstract.** Research on (Decentralized) Multi-Client Functional Encryption (or (D)MCFE) is very active, with interesting constructions, especially for the class of inner products. However, the security notions have been evolving over the time. While the target of the adversary in distinguishing ciphertexts is clear, legitimate scenarios that do not consist of trivial attacks on the functionality are less obvious. In this paper, we wonder whether only trivial attacks are excluded from previous security games. And, unfortunately, this was not the case.

We then propose a stronger security notion, with a large definition of admissible attacks, and prove it is optimal: any extension of the set of admissible attacks is actually a trivial attack on the functionality, and not against the specific scheme. In addition, we show that all the previous constructions are insecure w.r.t. this new security notion. Eventually, we propose new DMCFE schemes for the class of inner products that provide the new features and achieve this stronger security notion.

**Keywords:** Functional Encryption, Corruptions, Security Notions

## 1 Introduction

**Decentralized Multi-Client Functional Encryption.** Multi-Input Functional Encryption (MIFE) and Multi-Client Functional Encryption (MCFE), together with their decentralized variants [GGG<sup>+</sup>14, GKL<sup>+</sup>13, CDG<sup>+</sup>18a], have been receiving a strong interest from the cryptographic community. They generalize the nice functional encryption primitive [SW05, BSW11] where the single input  $x$ , in the encryption procedure, is split into an input vector  $(x_1, \dots, x_n)$ , and the components can be encrypted independently, possibly by different senders/clients in MCFE. An index  $i$  for each component, and a (typically time-based) tag  $\text{tag}$  for MCFE, are used for every encryption  $c_i = \text{Enc}(i, \text{tag}, x_i)$ . From the  $n$  encrypted components under the same tag  $\text{tag}$ , anyone owning a functional decryption key  $\text{dk}_f$ , for the  $n$ -ary function  $f$ , can compute  $f(x_1, \dots, x_n)$  but nothing else about the individual  $x_i$ 's. In this paper, we focus on a standard and optimal security model for the most general form of MCFE, namely *decentralized* MCFE, where the generation of functional decryption keys is also split between multiple clients.

**Previous Corruption Model for (D)MCFE.** In previous (D)MCFE, encryption was claimed to require a private key  $\text{ek}_i$  per client, for each component  $c_i$ , because of deterministic encryption. Then, some of these keys might get corrupted. In DMCFE, where multiple clients contribute to generate the decryption functional keys and also own secret keys  $\text{sk}_i$ , and some can get corrupted. Therefore, there exists potential corruption of two categories of keys regarding DMCFE that need to be dealt with: a private encryption key  $\text{ek}_i$  for encryption and a secret key  $\text{sk}_i$  for generating functional keys. The proposed corruption model in the work on DMCFE by Chotard *et al.* [CDG<sup>+</sup>18a] is: when an adversary corrupts a client  $i$ , it receives both  $(\text{sk}_i, \text{ek}_i)$ . However, this does not reflect the real-life situation. In fact, the encryption keys  $\text{ek}_i$ 's and the secret keys  $\text{sk}_i$ 's can have different levels of protection ( $\text{sk}_i$  looks more critical than  $\text{ek}_i$ ) and can be stored on different devices. This is thus a strong restriction to get both keys in case of corruption. Actually, this corruption model was employed in the previous DMCFE constructions for inner products  $f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$ , as the numbers of  $\text{sk}_i$ 's and  $\text{ek}_i$ 's are equal, and in most of them particularly,  $\text{sk}_i$  is either included in  $\text{ek}_i$ , *e.g.* [CDG<sup>+</sup>18a, CDG<sup>+</sup>18b, LT19], or they are the

same, *e.g.* [ABKW19, ABG19, AGT21b]<sup>3</sup>. But this might not always be the case. Specifically, for quadratic functions computing  $f_{\mathbf{A}}(\mathbf{x}) = \mathbf{x}^{\top} \mathbf{A} \mathbf{x}$  as considered in [AGT21a, AGT22], one could have  $n^2$  secret keys  $\mathbf{sk}_j$  for the square matrix  $\mathbf{A}$  and  $n$  encryption keys  $\mathbf{ek}_i$  only for the input vector  $\mathbf{x}$ . Hence, the holders of  $\mathbf{sk}_j$ 's and  $\mathbf{ek}_i$ 's might differ.

**Previous Notions of Admissible Attacks against (D)MCFE.** Generally, studying an advanced cryptographic primitive involves formalizing the ubiquitous perception of trivial attacks when devising its security notion, those that exploit only the *functionality* of the primitive to trivially break any specific constructions. A standard example is the case of *identity-based encryption* [Sha84, Coc01, BF01, BGH07], of which the widely agreed security notion forbids the adversary to obtain the secret key of any identity that could decrypt the challenge ciphertext. In our case of (D)MCFE, everything becomes much more complicated due to the computational aspect of the function class and the corruption in multi-user setting. Following the introduction of (D)MCFE in the seminal paper [CDG<sup>+</sup>18a], to the best of our knowledge, all follow-up studies on (D)MCFE, for instance [CDG<sup>+</sup>18b, ABKW19, ABG19, LT19, CDSG<sup>+</sup>20, AGT21b], administered an *admissibility condition* in order to prohibit trivial attacks, and restricted particularly adversaries to asking the challenge components  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$  in case of a corrupted  $i$ . Attacks that satisfy the admissibility condition are called *admissible attacks*. Nonetheless, there was no satisfactory justification for such a restriction, except that all the constructions used a deterministic encryption, and so the corruption of  $\mathbf{ek}_i$  could allow to re-encrypt  $\mathbf{x}_0^*[i]$  and compare with the challenge ciphertext. This was thus also the similar argument to support private encryption keys. Since then, relaxing the foregoing constraint was widely believed to be insurmountable for constructing (D)MCFE schemes and proving their security.

**An Improved Security Model for DMCFE.** Since previous security notions of DMCFE turn out unstable, the main goal of this paper is to propose a fair and optimal security model. SEPARATING CORRUPTIONS OF  $\mathbf{ek}_i$  AND  $\mathbf{sk}_i$ . Our first step is thus to *separate* the corruption of  $\mathbf{sk}_j$  from that of  $\mathbf{ek}_i$ , *i.e.* the adversary must specify which type of keys it wants to corrupt. This gives more flexibility to the adversary. However, its goal remains the same: distinguish between the encryption of  $\mathbf{x}_0^*$  and  $\mathbf{x}_1^*$  in the challenge ciphertext. We notice that this new corruption model captures the previous “both-or-nothing” model in previous works and any scheme that is secure in this new fine-grained model will also be secure in the old one. A very recent work by Agrawal *et al.* [AGT21b] also defined a security model with similar fine-grained corruption, though as mentioned earlier (see footnote 3) their subsequent DMCFE scheme for inner products has  $\mathbf{sk}_i = \mathbf{ek}_i$  for every  $i$  and by corrupting one an adversary will obtain both keys.

REFINING ADMISSIBILITY FOR A STRONGER SECURITY. Our next objective consists in challenging the belief from previous admissibility conditions and relaxing the restriction  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$  in case of a corrupted  $i$ . A more relaxed admissibility means more attacks will be considered, leading to a stronger security notion. To summarize, we revise the security model for (D)MCFE and

1. We provide a new security model for DMCFE under separated corruption of keys and less restrictive admissibility condition. Our security model covers the security model in all previous works, in the sense that being secure in the former implies being secure in the latter.

In Section 3.1, we give the intuition of our new formulation for admissibility condition. This new definition will require probabilistic encryption, which exclude the need of private encryption keys. Our security model will thus also consider public-key encryption, as some security still holds when all the encryption keys  $\mathbf{ek}_i$  are corrupted. Note however this might make sense for

<sup>3</sup> The work of [AGT21b] constructs *function-hiding dynamic decentralized FE*, which directly yields a DMCFE with a stronger property of function secrecy. Even though their proposed security model captures separated corruption of  $\mathbf{ek}_i$  and  $\mathbf{sk}_i$ , implying they are different, their dynamic decentralized FE construction uses the same key for both and so does the resulting DMCFE, *i.e.*  $\mathbf{sk}_i = \mathbf{ek}_i$  for every  $i$ .

limited classes of functions only and becomes completely meaningless when both  $(ek_i, sk_i)$  can be corrupted at once.

**Optimality.** At the core of our new security model is a more relaxed admissibility condition. Up to this point one may well wonder if there is still room to relax our condition, in the same way we have done to the admissibility condition put forth since the birth of DMCFE in [CDG<sup>+</sup>18a]. Our goal is to analyze this question in a rigorous manner. This turns out to be notoriously hard because we aim to settle this infamous problem with satisfactory justifications whenever a new condition is introduced. Intuitively, since all prior works did not elaborate formally whether an admissibility condition must be respected or it is just optional, we have to start from scratch to formalize how “indispensable” a condition is. We thus address this *optimality* question and this leads to our second contribution:

2. We provide a new framework to prove the optimality of our new notion of *admissible attacks*. More formally, this allows us to show that any non-admissible attack would actually break any efficient construction for the functionality. This proves that we only exclude attacks that are at the functionality level and not at the scheme level.

We believe that the conceptual message from our methodology is one main contribution of this paper. We refer to Section 3.3 for a detailed explanation of our modeling choices as well as the encountered problems.

**Impact and Feasibility.** While we have shown our security notion to be optimal w.r.t. the functionality for a class of functions, there are two remaining questions, with respect to this new admissibility notion: are the previous constructions secure? Can we construct concrete schemes for non-trivial functionalities?

First, we can show that the class of inner products is a non-trivial class. Furthermore, it has been widely studied, with several candidates: the DDH-based MCFE for inner products from [CDG<sup>+</sup>18a, ABG19, CDSG<sup>+</sup>20] cannot be proven secure in our model, due to the following attack, which was artificially excluded in the previous security models. For any corrupted key  $ek_i$ , it was required that  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$ , because of the deterministic encryption: an adversary corrupts client 1 among  $n$  clients to get  $ek_1$ , then queries the function  $\mathbf{y}$  with  $\mathbf{y}[1] = 0$  and challenges  $(\mathbf{x}_0^*[i], \mathbf{x}_1^*[i])_{i \in [n]}$  such that  $\mathbf{x}_0^*[1] \neq \mathbf{x}_1^*[1]$  and  $\langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle$ . Then, the adversary encrypts  $\mathbf{x}_1^*[1]$  on their own using  $ek_1$ . By comparing with the obtained ciphertext on  $\mathbf{x}_b^*[1]$ , such an adversary can decide correctly on  $b$ . In addition to these DDH-based constructions, in a work by Libert and Titu [LT19], the authors proposed the first LWE-based MCFE in the standard model. The ciphertext components of this scheme is somewhat randomized by some small Gaussian error, but the above attack still works by choosing  $\mathbf{x}_0^*[1] \neq \mathbf{x}_1^*[1]$  that are far from each other, then deciding based on the norm of the two ciphertexts’ difference<sup>4</sup>. We note that the above attack gives a byproduct that complements our first contribution

- 1-bis. Our security model is strictly stronger than the security model in almost all previous works, in the sense that prior concrete schemes cannot be proven secure in ours.

Besides the theoretical part introducing and proving our optimal security notion for DMCFE, we also propose new constructions in the DDH setting which meet the proposed level of security. This requires a number of new technical ideas, in particular a technique for achieving admissibility *via revocation* (in a different way than [ABP<sup>+</sup>17]) and using *dual pairing vector spaces* (DPVS) [OT10, OT12a, OT12b], to build a probabilistic encryption scheme.

Roughly speaking, our new admissibility when translated for the particular cases of inner-products introduces one condition that for all corrupted clients  $i$ , for  $ek_i$ , for all functional key query  $\mathbf{y}$ , it must hold that

$$(\mathbf{x}_0^*[i] - \mathbf{x}_1^*[i]) \cdot \mathbf{y}[i] = 0 . \quad (1)$$

<sup>4</sup> We use the metrics employed in the context of the LWE-based (D)MCFE in [LT19].

Previous security models required  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$ , but we now have to deal with the case  $\mathbf{x}_0^*[i] \neq \mathbf{x}_1^*[i]$  and  $\mathbf{y}[i] = 0$  additionally. A necessary condition is that our encryption must be probabilistic (otherwise, the attack described in the previous paragraph applies). However, that is *not* enough because we want semantic security for the ciphertext component  $\text{ct}_i$  of  $\mathbf{x}_b^*[i]$  as well, where  $b \stackrel{\$}{\leftarrow} \{0, 1\}$  is the challenge bit. When we view this problem under the lens of revocation systems, similarities emerge: as soon as the special value 0 is set for  $\mathbf{y}[i]$ , we want to nullify the ability for recovering information about  $\mathbf{x}_b^*[i]$ . The foregoing fits well in the context of revocation. Conveniently, the work by Agrawal *et al.* [ABP<sup>+</sup>17] solved the “dual” problem, namely using IPFE to construct revocation systems, and along the way, the authors of [ABP<sup>+</sup>17] presented a DDH-based IPFE that we can embed locally into the vectors in DPVS, components by components. We leverage this idea to concoct DPVS-based DMCFE schemes for inner-product functionality and achieve security under the condition (1). In the end, our third contribution is

3. We demonstrate the feasibility of our new security model by presenting DDH-based DMCFE schemes for inner products over polynomially bounded ranges using pairings, the first concrete scheme whose security holds against fine-grained corruptions and a less restrictive admissibility.

More high-level details are provided in Section 3.4.

## 2 Preliminaries

We write  $[n]$  to denote the set  $\{1, 2, \dots, n\}$  for an integer  $n$ . For any  $q \geq 2$ , we let  $\mathbb{Z}_q$  denote the ring of integers with addition and multiplication modulo  $q$ . For a prime  $q$  and an integer  $N$ , we denote by  $GL_N(\mathbb{Z}_q)$  the general linear group of degree  $N$  over  $\mathbb{Z}_q$ . We write vectors as row-vectors, unless stated otherwise. For a vector  $\mathbf{x}$  of dimension  $n$ , the notation  $\mathbf{x}[i]$  indicates the  $i$ -th coordinate of  $\mathbf{x}$ , for  $i \in [n]$ . We will follow the implicit notation in [EHK<sup>+</sup>13] and use  $\llbracket a \rrbracket$  to denote  $g^a$  in a cyclic group  $\mathbb{G}$  of prime order  $q$  generated by  $g$ , given  $a \in \mathbb{Z}_q$ . This implicit notation extends to matrices and vectors having entries in  $\mathbb{Z}_q$ . We use the shorthand **ppt** for “probabilistic polynomial time”. In the security proofs, whenever we use an ordered sequence of games  $(\mathbb{G}_0, \mathbb{G}_1, \dots, \mathbb{G}_i, \dots, \mathbb{G}_L)$  indexed by  $i \in \{0, 1, \dots, L\}$ , we refer to the predecessor of  $\mathbb{G}_j$  by  $\mathbb{G}_{j-1}$ , for  $j \in [L]$ .

### 2.1 Hardness Assumptions

We state the assumptions needed for our constructions.

**Definition 1.** *In a cyclic group  $\mathbb{G}$  of prime order  $q$ , the **Decisional Diffie-Hellman** (DDH) problem is to distinguish the distributions*

$$D_0 = \{(\llbracket 1 \rrbracket, \llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket ab \rrbracket)\} \quad D_1 = \{(\llbracket 1 \rrbracket, \llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket c \rrbracket)\}.$$

for  $a, b, c \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ . The DDH assumption in  $\mathbb{G}$  assumes that no **ppt** adversary can decide the DDH problem with non-negligible probability.

In the bilinear setting  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ , the **Symmetric eXternal Diffie-Hellman** (SXDH) assumption makes the DDH assumption in both  $\mathbb{G}_1$  and  $\mathbb{G}_2$ .

**Definition 2.** *In the bilinear setting  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ , the **Decisional Bilinear Diffie-Hellman** (DBDH) problem is to distinguish the distributions*

$$D_0 = \{(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2, \llbracket abc \rrbracket_t)\} \quad D_1 = \{(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2, \llbracket r \rrbracket_t)\}.$$

for  $a, b, c, r \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ . The DBDH assumption in  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$  assumes that no **ppt** adversary can decide the DBDH problem with non-negligible probability.

## 2.2 Dual Pairing Vector Spaces

Our constructions rely on the *Dual Pairing Vector Spaces* (DPVS) framework in prime-order bilinear group setting  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$  and  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$  are all written additively. The DPVS technique dates back to the seminal work by Okamoto-Takashima [OT10, OT12a, OT12b] aiming at adaptive security for ABE as well as IBE, together with the *dual system methodology* introduced by Waters [Wat09]. In [LW10], the setting for dual systems is composite-order bilinear groups. Continuing on this line of works, Chen *et al.* [CLL<sup>+</sup>13] used prime-order bilinear groups under the SXDH assumption. Let us fix  $N \in \mathbb{N}$  and consider  $\mathbb{G}_1^N$  having  $N$  copies of  $\mathbb{G}_1$ . Any  $\mathbf{x} = \llbracket (x_1, \dots, x_N) \rrbracket_1 \in \mathbb{G}_1^N$  is identified as the vector  $(x_1, \dots, x_N) \in \mathbb{Z}_q^N$ . There is no ambiguity because  $\mathbb{G}_1$  is a cyclic group of prime order  $q$ . The  $\mathbf{0}$ -vector is  $\mathbf{0} = \llbracket (0, \dots, 0) \rrbracket_1$ . The addition of two vectors in  $\mathbb{G}_1^N$  is defined by coordinate-wise addition. The scalar multiplication of a vector is defined by  $t \cdot \mathbf{x} := \llbracket t \cdot (x_1, \dots, x_N) \rrbracket_1$ , where  $t \in \mathbb{Z}_q$  and  $\mathbf{x} = \llbracket (x_1, \dots, x_N) \rrbracket_1$ . The additive inverse of  $\mathbf{x} \in \mathbb{G}_1^N$  is defined to be  $-\mathbf{x} := \llbracket (-x_1, \dots, -x_N) \rrbracket_1$ . 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  $\llbracket B^{(i)} \rrbracket_1$ , where  $B^{(i)}$  is the  $i$ -th row of  $B$ . The canonical basis  $\mathbf{A}$  of  $\mathbb{G}_1^N$  consists of  $\mathbf{a}_1 := \llbracket (1, 0, \dots, 0) \rrbracket_1, \mathbf{a}_2 := \llbracket (0, 1, 0, \dots, 0) \rrbracket_1, \dots, \mathbf{a}_N := \llbracket (0, \dots, 0, 1) \rrbracket_1$ . It is straightforward that we can write  $\mathbf{B} = B \cdot \mathbf{A}$  for any basis  $\mathbf{B}$  of  $\mathbb{G}_1^N$  corresponding to an invertible matrix  $B \in GL_N(\mathbb{Z}_q)$ , for the change of basis. We write  $\mathbf{x} = (x_1, \dots, x_N)_{\mathbf{B}}$  to indicate the representation of  $\mathbf{x}$  in the basis  $\mathbf{B}$ , i.e.  $\mathbf{x} = \sum_{i=1}^N x_i \cdot \mathbf{b}_i$ . 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  $B' := (B^{-1})^\top$  and the  $i$ -th row  $\mathbf{b}_i^*$  of  $\mathbf{B}^*$  is  $\llbracket B'^{(i)} \rrbracket_2$ . It holds that  $B \cdot (B')^\top = I_N$  the identity matrix and  $\mathbf{b}_i \times \mathbf{b}_j^* = \llbracket \delta_{i,j} \rrbracket_t$  for every  $i, j \in [N]$ , where  $\delta_{i,j} = 1$  if and only if  $i = j$ . Treating  $\mathbb{G}_2^N$  similarly, we can furthermore define a product of two vectors  $\mathbf{x} = \llbracket (x_1, \dots, x_N) \rrbracket_1 \in \mathbb{G}_1^N, \mathbf{y} = \llbracket (y_1, \dots, y_N) \rrbracket_2 \in \mathbb{G}_2^N$  by  $\mathbf{x} \times \mathbf{y} := \prod_{i=1}^N \mathbf{e}(\mathbf{x}[i], \mathbf{y}[i]) = \llbracket \langle (x_1, \dots, x_N), (y_1, \dots, y_N) \rangle \rrbracket_t$ . We call the pair  $(\mathbf{B}, \mathbf{B}^*)$  *dual orthogonal bases* of  $(\mathbb{G}_1^N, \mathbb{G}_2^N)$ . If  $\mathbf{B}$  is constructed by a random invertible matrix  $B \xleftarrow{\$} GL_N(\mathbb{Z}_q)$ , we call the resulting  $(\mathbf{B}, \mathbf{B}^*)$  a pair of random dual bases. A DPVS is a bilinear group setting  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q, N)$  with dual orthogonal bases. In this work, we also use extensively *basis changes* over dual orthogonal bases, the details are recalled in the appendix A.2.

## 3 Technical Overview

### 3.1 Motivations for a Refinement on Admissibility

In the seminal work on DMCFE for a function class  $\mathcal{F}$  by Chotard *et al.* [CDG<sup>+</sup>18a], the authors define the security game with oracles

**(Initialize, Corrupt, LoR, DKeyGenShare, Enc, Finalize)**

between a challenger and an adversary. The oracle **Initialize** sets up the parameters, including the number of clients  $n$  and their secret-encryption key pairs  $(\text{sk}_i, \text{ek}_i)$ . The oracle **DKeyGenShare** produces functional key components for  $F \in \mathcal{F}$  using  $\text{sk}_i$  under some function tag  $\text{tag-f} \in \text{Tag}$ , while **LoR** is the left-or-right oracle, which outputs the challenge ciphertext component of  $\mathbf{x}_b^*[i]$  under  $\text{tag} \in \text{Tag}$  upon receiving  $(\mathbf{x}_0^*[i], \mathbf{x}_1^*[i])$  for  $b \xleftarrow{\$} \{0, 1\}$ . An adversary can corrupt any client  $i$  of his choice by querying **Corrupt** so as to receive *both*  $(\text{sk}_i, \text{ek}_i)$ .

In the end, a set of conditions is specified such that the adversary wins the game only when they conform to these conditions and outputs a correct  $b'$  equal to the challenge bit  $b$ . The main reason there are such conditions is to focus only on the scenarios where a notion of semantic security really makes sense in this DMCFE setting. In this paper we call an attack that satisfies such conditions an *admissible* attack. Checking these conditions is done by a **Finalize** procedure in the security game, according to the sets  $\mathcal{C}$  of corrupted clients (asked to **Corrupt**),  $\mathcal{H}$  of

honest clients, and  $\mathcal{Q}$  of key queries (asked to **DKeyGenShare**) during the attack. To recall from [CDG<sup>+</sup>18a, Def. 2, Def. 5], the adversary’s output is ignored and replaced by a random bit, *i.e.* the attack is NOT admissible, if one of the following holds:

1. There exists a corrupted client  $i \in \mathcal{C}$  such that the  $i$ -th components of the challenge messages  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  are not the same, *i.e.*  $\mathbf{x}_0^*[i] \neq \mathbf{x}_1^*[i]$ .
2. There exists a tag  $\text{tag} \in \text{Tag}$  (respectively,  $\text{tag-f} \in \text{Tag}$ ) and  $i \neq j \in \mathcal{H}$  such that the  $j$ -th challenge component (respectively, key component) is queried but the  $i$ -th challenge component (respectively, key component) is not.
3. None of the two above conditions are satisfied, but there exists a function  $F$  queried to **DKeyGenShare** that differs on  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$ , *i.e.*  $F(\mathbf{x}_0^*) \neq F(\mathbf{x}_1^*)$ .

Our observation is that only the condition 3 can be justified for the sake of avoiding trivial attacks, while the other conditions 1 and 2 do not have satisfactory explanations. About condition 1, we have seen from the attacks in the paragraph **Impact and Feasibility** of Section 1 that this condition is artificial and unfortunately excludes also non-trivial attacks. About condition 2, follow-up works [CDG<sup>+</sup>18b, CDSG<sup>+</sup>20] and other results on the subject [AGRW17, DOT18, ABKW19, ABG19] show that we can completely remove this constraint. Our objective now becomes devising a less restrictive admissible condition, which should capture and generalize only condition 3. We recall that a less restrictive condition implies *more* attacks will be considered non-trivial and we obtain a *stronger* security model.

### 3.2 Towards a New Admissibility Condition under Separated Corruption of Keys

Our first step is to *separate* the corruption of  $\text{sk}_i$  from that of  $\text{ek}_i$ , *i.e.* the adversary has to specify which type of keys with respect to component  $i$  he wants to corrupt. All prior works allow the adversary to corrupt both keys *at once*. This separation helps us define in a finer way which information the adversary can deduce using each type of corrupted keys, and thus even deal with public-key encryption. As a result, we have sets of corrupted and honest clients  $(\mathcal{C}_{\text{sk}}, \mathcal{H}_{\text{sk}}), (\mathcal{C}_{\text{ek}}, \mathcal{H}_{\text{ek}})$ , independently for the secret keys  $(\text{sk}_j)_j$  and the encryption keys  $(\text{ek}_i)_i$ . This even allows to have independent sets of clients owning the secret keys  $(\text{sk}_j)_j$  and the encryption keys  $(\text{ek}_i)_i$ . Our complete security experiment can be found in Figure 1. Being already mentioned in **Previous Corruption Model for (D)MCFE** of Section 1, to the best of our knowledge, almost all prior proposed constructions of (D)MCFE can not handle separate corruption of  $\text{ek}_i$  and  $\text{sk}_i$ , for example, see [CDG<sup>+</sup>18a, CDG<sup>+</sup>18b, LT19, ABKW19, ABG19, AGT21b], despite the fact that a such separation is meaningful and is indeed discussed notably in the security model of [AGT21b].

Next, we represent an  $n$ -ary function  $F : \mathcal{D}_1 \times \dots \times \mathcal{D}_n \rightarrow \mathcal{R}$  of a function class  $\mathcal{F}$  by a length- $m$  vector of parameters from  $\text{Param}_1 \times \dots \times \text{Param}_m$ , given by a deterministic encoding  $\mathbf{p} : \mathcal{F} \rightarrow \text{Param}_1 \times \dots \times \text{Param}_m$  and  $m$  can be different from  $n$ . Given such representations as vectors for both inputs and functions, we define the notion *deducible inputs and functions* (see Definition 3). More specifically, let  $\mathcal{H}_{\text{inp}} \subseteq [n], \mathcal{H}_{\text{func}} \subseteq [m]$  and suppose we are given  $\mathbf{x}_{\text{inp}} \in (\mathcal{D}_i)_{i \in \mathcal{H}_{\text{inp}}}$  and  $\mathbf{y}_{\text{func}} \in (\text{Param}_i)_{i \in \mathcal{H}_{\text{func}}}$  as lists of inputs and parameters that are indexed by  $\mathcal{H}_{\text{inp}}, \mathcal{H}_{\text{func}}$  respectively. A vector  $\mathbf{z}$  is *deducible* from  $\mathbf{x}_{\text{inp}}$  if their coordinates at positions in  $\mathcal{H}_{\text{inp}}$  are the same. Similarly, a function  $G$  is *deducible* from  $\mathbf{y}_{\text{func}}$  if its parameters coincide at positions in  $\mathcal{H}_{\text{func}}$  with  $\mathbf{y}_{\text{func}}$ . Intuitively, the lists  $(\mathbf{x}_{\text{inp}}, \mathbf{y}_{\text{func}})$  play the role of “honest” predetermined input’s components and function’s parameters, whilst the deducible  $(\mathbf{z}, G)$  signifies what the adversary can infer by manipulating the remaining “corrupted” parts of the input and function.

Being equipped with this notion of deducible inputs and functions, our admissible condition is formulated as:

Given the sets  $(\mathcal{H}_{\text{sk}}, \mathcal{H}_{\text{ek}})$ , an *attack* is NOT admissible if there exist  $\text{tag}, \text{tag-f} \in \text{Tag}$ , a function  $F \in \mathcal{F}$  with parameters  $\mathbf{y} = (y_j)_{j \in [m]}$ , two challenges  $(\mathbf{x}_0^*, \mathbf{x}_1^*) := (x_i^{(0)}, x_i^{(1)})_{i \in [n]}$

such that  $(F, \text{tag-f})$  is queried to **DKeyGenShare**,  $((x_i^{(0)}, x_i^{(1)})_{i \in [n]}, \text{tag})$  is queried to **LoR** and there exists a pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$ , a function  $G$  deducible from  $\mathbf{y}_{\text{skey}}$  satisfying  $G(\mathbf{z}^{(0)}) \neq G(\mathbf{z}^{(1)})$  where we define  $\mathbf{y}_{\text{skey}} := (y_i)_{i \in \mathcal{H}_{\text{skey}}}$  and for  $b \in \{0, 1\}$ ,  $\mathbf{x}_{\text{ekey}}^{(b)} := (x_i^{(b)})_{i \in \mathcal{H}_{\text{ekey}}}$ .

It can be verified that if an attack satisfies the previous notion of admissibility in the original work [CDG<sup>+</sup>18a], such an attack will satisfy our notion of admissibility as well. Moreover, we can adapt naturally our admissibility from DMCFE to MCFE<sup>5</sup> and also demonstrate that the prior DDH-based constructions for MCFE with deterministic encryption, for example from [CDG<sup>+</sup>18a, ABG19, CDSG<sup>+</sup>20] to name a few, as well as an LWE-based construction for MCFE from [LT19] with slightly randomized encryption by Gaussian errors, cannot be proven secure in our new model by giving a concrete *admissible* attack, as already explained in Section 1.

### 3.3 Optimality of the New Admissibility: A Conceptual Challenge

After formulating a new admissibility condition, one natural question arises: *Is this the most suitable condition?* From a conceptual point of view, we want to prove that

For *certain* function classes, our admissibility cannot be relaxed, *i.e.* one cannot admit *some* non-admissible attack following our definition and still hope to be able to construct *some* specific efficient scheme that is provably secure.

Unsurprisingly, this poses a great definitional problem.

First of all, in all previous studies on (D)MCFE starting from [CDG<sup>+</sup>18a], the admissibility concerns *adversaries* in the security game. Hence, if we want to prove the above claim, we need to consider *all* possible adversaries that can run non-admissible attacks and argue that they must be excluded. This is hard to argue rigorously, for example, what happens if a “dummy” adversary behaves in a non-admissible way but in the end outputs only a random guess for the challenge bit? Therefore, our very first step is to define the admissibility condition differently and take into account general *attacks* instead of adversaries. Afterwards, our optimality notion for an admissibility condition on attacks is stated that:

An admissibility is optimal for  $\mathcal{F}$  if we can construct a *passive* ppt distinguisher  $\mathcal{S}$  that receives *some* non-admissible attack coming from the queries of an adversary  $\mathcal{A}$  to a challenger **Chall** in the game for a DMCFE  $\mathcal{E}$ , uses only properties of  $\mathcal{F}$ , and devises a *generic* strategy to output the correct challenge bit with significant probability in the security game against *any* arbitrary DMCFE  $\mathcal{E}'$ .

Intuitively,  $\mathcal{S}$  passively observes the non-admissible queries in the attack from some specific interaction between  $\mathcal{A}$  against some specific scheme  $\mathcal{E}$ . Yet, these queries helps  $\mathcal{S}$  come up with a general approach to win significantly against any DMCFE scheme in a game that allows such non-admissible behaviors. This means it is impossible to prove security whenever this kind of behaviors is allowed. We formalize all these details in Definition 11 and Theorem 12, **Remark 14** elaborates more on the proof of our optimality claim. The the appendix B.1 verifies our admissibility’s optimality for concrete most-studied function classes. Informally, we will explain in Section 4 the framework we proposed for arguing the optimality of an admissibility. Finally, the detailed admissibility condition for the class of inner products is given in **Remark 15**.

### 3.4 DMCFE for Inner Products with Refined Security Model

After introducing a new notion of admissibility in the security model for DMCFE and argue its optimality, we provide concrete constructions of DMCFE for inner products that are secure in

<sup>5</sup> The admissibility for MCFE is the particular condition when  $\mathcal{H}_{\text{skey}} = [m]$  and thus  $\mathbf{y}_{\text{skey}} = \mathbf{y}$ , meaning the only deducible function is  $F$  itself.

this model. Our results are twofold. In Section 5.1 we give an intermediate construction where the new admissibility is translated in the case of inner-products together with the *complete queries restriction* (similar to condition 2 in previous works). In Section 6, we leverage this backbone construction from Section 5.1 to remove this complete queries restriction via a generic transformation as well as a concrete scheme with improved security.

In the following we highlight the main ideas of our backbone construction in Section 5.1. Our function class of interest is for computing inner products  $\mathcal{F}^{\text{IP}} = \{F_{\mathbf{y}}\}$  and  $F_{\mathbf{y}} : (\mathbb{Z}_q^*)^n \rightarrow \mathbb{Z}_q$  is defined as  $F_{\mathbf{y}}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle$ . The parameter vector of  $F_{\mathbf{y}}$  is simply  $\mathbf{y} = (y_1, \dots, y_n) \in \mathbb{Z}_q^n$  and thus the number of parameters is the same as the dimension of the  $\mathbb{Z}_q$ -vector space for a prime  $q$ . Our construction relies on the notion of *Dual Pairing Vector Spaces* (DPVSeS, see Section 2.2). We use DPVSeS in the (additively written) bilinear group setting  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \mathbf{e}, g_1, g_2, g_t)$ . We sample  $n$  pairs of random dual bases  $(\mathbf{H}_i, \mathbf{H}_i^*)_{i=1}^n$ . Each client  $i$  will use their encryption key  $\text{ek}_i$  to encrypt the component  $x_i$  under some  $\text{tag}$  to get a ciphertext component  $\text{ct}_{\text{tag},i}$ , which is a vector of elements in  $\mathbb{G}_1$  computed using  $\mathbf{H}_i$ . Accordingly, each secret key  $\text{sk}_i$  will be used by the DKShare in the decentralized key generation so as to generate a key component  $\text{dk}_{\text{tag-f},i}$  for  $y_i$  under some  $\text{tag-f}$ , which is a vector of elements in  $\mathbb{G}_2$  computed using  $\mathbf{H}_i^*$ . During decryption, the product  $\text{ct}_{\text{tag},i} \times \text{dk}_{\text{tag-f},i}$  of vectors lying in dual bases will yield an element in  $\mathbb{G}_t$  of the form in the IPFE scheme by Agrawal, Libert, and Stehlé (ALS) [ALS16]. We denote by  $S = (s_1, \dots, s_n), U = (u_1, \dots, u_n)$  two vectors of secret scalars, intuitively which will be used in ALS ciphertext components  $\llbracket s_i \omega + u_i \omega' + x_i \rrbracket$ , where  $\llbracket (\omega, \omega') \rrbracket \leftarrow \mathbf{H}(\text{tag})$  comes from a full-domain hash function. In a centralized setting, such as the single-client scheme in [ALS16] or the MCFE scheme in [CDG<sup>+</sup>18a], the ALS key extraction provides  $\langle S, \mathbf{y} \rangle$  and  $\langle U, \mathbf{y} \rangle$  to be used in decryption.

**Decentralizing ALS Key Extraction under Separated Corruption.** The first technical challenge is how to implement the ALS key extraction in a decentralized manner, because each key generator possessing  $y_i$  will not be able to compute  $\langle S, \mathbf{y} \rangle$  and  $\langle U, \mathbf{y} \rangle$  due to the lack of  $(y_j)_{j \neq i}$ . Our idea is to use  $(s_i y_i, u_i y_i)$  in the  $i$ -th key components, masked by some randomness, then exploit the properties of products in DPVS that multiply facing coordinates together in order to “glue” this randomness to appropriate values in the  $i$ -th ciphertext component that enables correct decryption. More specifically, the components have the following form:

$$\begin{array}{l} \text{ct}_{\text{tag},i} \left( \dots \left| \begin{array}{c} \omega p_i \\ s_i y_i \alpha_i + u_i y_i \gamma'_i \end{array} \right| \begin{array}{c} \omega' q_i \\ s_i y_i \gamma_i + u_i y_i \alpha_i \end{array} \right| \text{ALS-ciph} \left| \dots \right)_{\mathbf{H}_i}; \\ \text{dk}_{\text{tag-f},i} \left( \dots \left| \begin{array}{c} \omega p_i \\ s_i y_i \alpha_i + u_i y_i \gamma'_i \end{array} \right| \begin{array}{c} \omega' q_i \\ s_i y_i \gamma_i + u_i y_i \alpha_i \end{array} \right| y_i \left| \dots \right)_{\mathbf{H}_i^*}; \end{array}$$

where ALS-ciph is the scalar in ALS ciphertext and  $(p_i, q_i)$  in  $\text{ct}_{\text{tag},i}$  together with  $(\alpha_i, \gamma_i, \gamma'_i)$  in  $\text{dk}_{\text{tag-f},i}$  satisfy  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4$  and  $\zeta_1, \zeta_2, \zeta_3, \zeta_4 \xleftarrow{\$} \mathbb{Z}_q^*$  are determined at setup time. However, the aforementioned local gluing technique is not enough, as it alone still reveals information about individual  $x_i y_i$  from  $\text{ct}_{\text{tag},i} \times \text{dk}_{\text{tag-f},i}$ . A remedy is to put another layer of random secret sharings  $\theta_i \xleftarrow{\$} \mathbb{Z}_q$  into the partial key components such that  $\sum_{i=1}^n \theta_i = 0$  so that only when  $n$  pairs of ciphertext-key components are combined together will we obtain the decrypted result. This new secret shares  $(\theta_i)_i$  are randomized by  $d_{\text{tag-f}} \leftarrow \mathbf{H}(\text{tag-f})$  for each functional key, the newly randomized  $(d_{\text{tag-f}} \theta_i)_i$  will behave indistinguishably from a random independent secret sharing of 0 under DDH. We refer to Section 5.1 and the transition  $\mathbb{G}_6 \rightarrow \mathbb{G}_7$  in the proof of Theorem 16 for more details.

**HANDLING SEPARATED CORRUPTION.** Each encryption key  $\text{ek}_i$  will contain information relevant to the basis  $\mathbf{H}_i$  so that client  $i$  can compute  $\text{ct}_{\text{tag},i}$ , meanwhile each key generator can use  $\text{sk}_i$  related to the *dual* basis  $\mathbf{H}_i^*$  for deriving the partial  $\mathbf{k}_{i,\text{ipfe}}$ . It appears that the contents of  $\text{ek}_i$  and  $\text{sk}_i$  belong to dual bases, independent for each  $i$ , and we handle their separated corruption by *basis changes* over  $(\mathbf{H}_i, \mathbf{H}_i^*)$  in DPVS. We note that as soon as we program the basis change of one basis, we cannot control the change on its dual counterpart (defined by linear relations, see the appendix A.2). To this end, our proofs can handle at best the scenario where one key type must be *statically* corrupted whereas the other’s corruption can be *dynamic*, otherwise the fact that for some  $i$  the keys  $\text{ek}_i$  and  $\text{sk}_i$  can be corrupted dynamically, in separate ways, can lead to inconsistency between basis changes. In particular, we use basis changes to program the



master secret values  $(s_i, u_i)_i$  as well as the secret sharings  $(\theta_i)_i$ , thus we want to program the changes on the dual bases  $\mathbf{H}_i^*$ . Consequently, we enforce static corruption on  $\text{sk}_i$  and perform those changes on  $i$  corresponding to honest  $\text{sk}_i$ <sup>6</sup>.

**Achieving New Admissibility by Embedding Revocation Mechanism into Components.** The second technical challenge is to handle our new admissibility. In the prior weaker admissible condition introduced in [CDG<sup>+</sup>18a], where  $(\text{ek}_i, \text{sk}_i)$  are corrupted together, if  $i \in [n]$  is corrupted then  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$ . Putting forward the translation of our new admissibility in the functionality for inner products, the concrete conditions are: let  $\Delta \mathbf{x}[i] := \mathbf{x}_0^*[i] - \mathbf{x}_1^*[i]$ , then

1. For all  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$ ,  $\sum_{i \in (\mathcal{H}_{\text{skey}} \cup \mathcal{H}_{\text{ekey}})} \Delta \mathbf{x}[i] \mathbf{y}[i] = 0$ .
2. For all  $i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ , either  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$  or  $\mathbf{y}[i] = 0$ .
3. For all  $i \in \mathcal{C}_{\text{skey}}$ ,  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$ .

As the main complication compared to [CDG<sup>+</sup>18a, CDSG<sup>+</sup>20], when  $i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$  it can be the case that  $\Delta \mathbf{x}[i] \neq 0$  and  $\mathbf{y}[i] = 0$ . We want to ensure that even in this configuration the adversary cannot distinguish the  $i$ -th ciphertext components.

We interpret this situation in the language of *revocation*: if the adversary obtains the  $i$ -th key component  $\text{dk}_{\text{tag-f}, i}$  for  $y_i := \mathbf{y}[i] = 0$ , which is honestly generated as  $i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ , then implicitly we are “revoking” the ability to learn information about the  $i$ -th challenge component using  $\text{dk}_{\text{tag-f}, i}$ , even when the adversary can encrypt whatsoever using the corrupted  $\text{ek}_i$ , whose role now resembles a “public key” as in usual revocation systems. This leads us to the idea of embedding some sort of DDH-based revocation technique into each  $i$ -th component. We need to apply some revocation technique that is compatible with the bilinear group setting *and* the ALS ciphertext form, which is current employ at each component  $i$  in  $\text{ct}_{\text{tag}, i}$ . We turn our attention to a recent work by Agrawal *et al.* [ABP<sup>+</sup>17], which builds public trace-and-revoke systems from standard assumptions and is particularly suitable for our objective because their constructions can be generically based on the DDH-based ALS IPFE. At a very high level, the decryption for  $m$  of the precedent scheme for revocation can be recasted as:

$$\frac{\text{ALS-IPFE.Dec}(\text{sk}_{id}, \text{ct})}{\langle \mathbf{x}_{id}, \mathbf{v}_R \rangle} = \frac{\langle \mathbf{x}_{id}, \mathbf{v}_R \cdot m \rangle}{\langle \mathbf{x}_{id}, \mathbf{v}_R \rangle} = m \quad (2)$$

where  $\text{sk}_{id}$  is the decryption given for an identity  $id$ ,  $\mathbf{x}_{id}$  is some vector associated to  $id$ , and  $\mathbf{v}_R$  is derived from the revoked set  $R$ . With overwhelming probability,  $\langle \mathbf{x}_{id}, \mathbf{v}_R \rangle \neq 0$  conditioned on  $id \notin R$ .

To adapt to our situation the division is translated to subtraction of coordinates and our “revoking” test depends only on a scalar  $y_i$  and whether  $y_i = 0$  or not, the inner product become scalar multiplications in  $\mathbb{Z}_q^*$ . Consequently, we introduce extra coordinates in the DPVS bases  $(\mathbf{H}_i, \mathbf{H}_i^*)$  to implement the aforementioned revocation technique, locally inside the vector’s components as follows:

$$\begin{array}{l} \text{ct}_{\text{tag}, i} \left( \begin{array}{c|c|c} \omega p_i & \omega' q_i & \text{ALS-ciph} - r_i v_i \\ \hline s_i y_i \alpha_i + u_i y_i \gamma_i & s_i y_i \gamma_i + u_i y_i \alpha_i & y_i \end{array} \middle| \begin{array}{c} r_i t_i \\ y_i v_i / t_i \\ \dots \\ d_{\text{tag-f}} \theta_i \end{array} \middle| \text{rand} \right)_{\mathbf{H}_i}; \\ \text{dk}_{\text{tag-f}, i} \left( \begin{array}{c|c|c} \omega p_i & \omega' q_i & \text{ALS-ciph} - r_i v_i \\ \hline s_i y_i \alpha_i + u_i y_i \gamma_i & s_i y_i \gamma_i + u_i y_i \alpha_i & y_i \end{array} \middle| \begin{array}{c} r_i t_i \\ y_i v_i / t_i \\ \dots \\ d_{\text{tag-f}} \theta_i \end{array} \middle| \text{rand} \right)_{\mathbf{H}_i^*}; \end{array}$$

Using extra coordinates to contain  $\llbracket (\text{ALS-ciph} - r_i v_i, r_i t_i) \rrbracket_1$  in  $\text{ct}_{\text{tag}, i}$  as well as  $\llbracket (y_i, y_i v_i / t_i) \rrbracket_2$  in  $\text{dk}_{\text{tag-f}, i}$  helps us perform a “local” ALS+revocation decryption for component  $i$ , following the idea (2), when performing  $\text{ct}_{\text{tag}, i} \times \text{dk}_{\text{tag-f}, i}$ . Intuitively, our uniformly random scalar  $r_i \xleftarrow{\$} \mathbb{Z}_q$  plays the role similar to that of  $\mathbf{x}_{id}$  in the blueprint from [ABP<sup>+</sup>17], that helps proving semantic security in the case of “revoked”  $y_i = 0$  under random basis changes in DPVS using DDH. This probabilistic layer with  $r_i \xleftarrow{\$} \mathbb{Z}_q$  allows to deal with corrupted encryption keys, even when  $\mathbf{x}_0^*[i] \neq \mathbf{x}_1^*[i]$ . This somehow covers public-key encryption. Our scheme is secure in the stronger security model under new admissibility and the complete queries restriction, *adaptively* in the challenges, with *dynamic* corruption of  $\text{ek}_i$  and *static* corruption of  $\text{sk}_i$ .

<sup>6</sup> There are further involved technicalities to ensure that  $\text{ek}_i$  is constructed consistently, *e.g.* see the transition  $G_7 \rightarrow G_8$  in the proof of Theorem 16.

## 4 A Stronger Security Model for Decentralized Multi-Client Functional Encryption

This section presents a new security model for (D)MCFE, in which we refine the *admissibility* of adversaries in the security game and allow a more fine-grained corruption of keys. Following the line of works by Chotard *et al.* [CDG<sup>+</sup>18a, CDSG<sup>+</sup>20], such a notion of admissible adversaries is for excluding the attacks that are inevitable under which we cannot prove security. Our objective is to define the admissible condition in a way that excludes as few attacks as possible, and as soon as such condition is weakened, there is an unconditional generic attack to trivially win the security game against *any* concrete scheme. First of all, Definition 3 formalizes the terminologies of parameters of a function and deducible functions/inputs.

**Definition 3 (Deducible inputs and functions).** Fix  $\lambda \in \mathbb{N}$  and denote by  $\mathcal{F}_\lambda$  a family of  $n$ -ary functions indexed by  $\lambda$ , with domain  $\mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n}$  and range  $\mathcal{R}_\lambda$ , where  $n = n(\lambda) \in \mathbb{N}$  is a function. Let  $\text{Param}_1, \dots, \text{Param}_m$  denote  $m$  sets of parameters for functions in  $\mathcal{F}_\lambda$ , where  $m = m(\lambda) \in \mathbb{N}$  is a function. Suppose there exists a deterministic encoding  $\mathbf{p} : \mathcal{F}_\lambda \rightarrow \text{Param}_1 \times \cdots \times \text{Param}_m$ , that maps a function  $F_{\mathbf{y}} \in \mathcal{F}_\lambda$  to its parameters  $\mathbf{y} \in \text{Param}_1 \times \cdots \times \text{Param}_m$ . Let  $\mathcal{H}_{\text{inp}} \subseteq [n]$ ,  $\mathcal{H}_{\text{func}} \subseteq [m]$  and suppose we are given  $\mathbf{x}_{\text{inp}} \in (\mathcal{D}_{\lambda,i})_{i \in \mathcal{H}_{\text{inp}}}$  and  $\mathbf{y}_{\text{func}} \in (\text{Param}_j)_{j \in \mathcal{H}_{\text{func}}}$  as lists of inputs and parameters that are indexed by  $\mathcal{H}_{\text{inp}}, \mathcal{H}_{\text{func}}$  respectively.

A vector  $\mathbf{z} \in \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n}$  is said to be deducible from  $\mathbf{x}_{\text{inp}}$  if  $\forall i \in \mathcal{H}_{\text{inp}} : \mathbf{z}[i] = \mathbf{x}_{\text{inp}}[i]$ . A function  $G$  is said to be deducible from  $\mathbf{y}_{\text{func}}$  if for all  $i \in \mathcal{H}_{\text{func}}$ , we have  $\mathbf{p}(G)[i] = \mathbf{y}_{\text{func}}[i]$ .

**Remark 4.** If  $\mathbf{y}_{\text{func}} = \mathbf{y}$ , for the parameter  $\mathbf{y}$  of some function  $F_{\mathbf{y}}$ , then the only function deducible from  $\mathbf{y}_{\text{func}}$  is  $F_{\mathbf{y}}$  itself. In the DMCFE setting, there will be situations with non-trivial  $\mathbf{y}_{\text{func}}$  where  $\mathcal{H}_{\text{func}} \subsetneq [m]$ . For concrete function classes in our construction, we focus on the class to compute inner products  $\mathcal{F}^{\text{IP}} = \{F_{\mathbf{y}}\}$  and  $F_{\mathbf{y}} : (\mathbb{Z}_q^*)^n \rightarrow \mathbb{Z}_q$  that is defined as  $F_{\mathbf{y}}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle$ . For inner products, the parameters of a function  $F_{\mathbf{y}} \in \mathcal{F}^{\text{IP}}$  can be precisely defined to be  $\mathbf{p}(F_{\mathbf{y}}) := \mathbf{y} \in \mathbb{Z}_q^n$  and the number of parameters  $m$  is equal to the number of arguments  $n$ . When  $\mathcal{F}_\lambda$  is clear from context, we omit the subscript  $\lambda$ .

We now recall the notion of *decentralized multi-client functional encryption* (DMCFE) whose syntax is given below.

**Definition 5 (Decentralized Multi-Client Functional Encryption).** A decentralized multi-client functional encryption (DMCFE) scheme for a function class  $\mathcal{F}$  consists of five algorithms

$$(\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$$

that are defined below:

**Setup( $1^\lambda$ ):** Given as input a security parameter  $\lambda$  and  $n = n(\lambda), m = m(\lambda)$ , generate in a possibly interactive manner  $n$  encryption keys  $(\text{ek}_i)_{i \in [n]}$  as well as  $m$  secret keys  $(\text{sk}_j)_{j \in [m]}$  where  $m, n : \mathbb{N} \rightarrow \mathbb{N}$  are functions.

**Enc( $\text{ek}_i, \text{tag}, x_i$ ):** Given as inputs an encryption key  $\text{ek}_i$ , a message  $x_i \in \mathcal{D}_{\lambda,i}$ , and a tag  $\text{tag}$ , output a ciphertext  $\text{ct}_{\text{tag},i}$ .

**DKShare( $\text{sk}_j, \text{tag-f}, y_j$ ):** Given a secret key  $\text{sk}_j$ , a tag  $\text{tag-f} \in \text{Tag}$ , and the  $j$ -th parameter  $y_j$ , output a partial functional key  $\text{dk}_{\text{tag-f},j}$ .

**DKeyComb( $(\text{dk}_{\text{tag-f},j})_{j \in [m]}, \text{tag-f}, F$ ):** Given a tag  $\text{tag-f}$  together with a function  $F$  and  $m$  partial functional keys  $\text{dk}_{\text{tag-f},j}$  for the parameters  $\mathbf{p}(F)$ , output the functional key  $\text{dk}_{\text{tag-f},F}$ .

**Dec( $\text{dk}_{\text{tag-f},F}, \mathbf{c}$ ):** Given the functional decryption key  $\text{dk}_{\text{tag-f},F}$  and a list of ciphertexts  $\mathbf{c} := (\text{ct}_{\text{tag},i})_{i=1}^n$  of length  $n$ , output an element in  $\mathcal{R}_\lambda$  or an invalid symbol  $\perp$ .

We make the assumption that all public parameters are included in the secret keys and the encryption keys, as well as the (partial) functional decryption key.

**Correctness.** We require that for sufficiently large  $\lambda \in \mathbb{N}$ , for all  $\text{tag}, \text{tag-f} \in \text{Tag}$ , for all  $F \in \mathcal{F}$ ,  $(x_i)_{i \in [n]} \in \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n}$  and

$$\begin{aligned} \text{sk}_j, (\text{ek}_i)_{i \in [n]} &\leftarrow \text{Setup}(1^\lambda); \text{dk}_{\text{tag-f},j} \leftarrow \text{DKShare}(\text{sk}_j, \text{tag-f}, y_j)_{j \in [m]} ; \\ \text{dk}_{\text{tag-f},F} &\leftarrow \text{DKeyComb}((\text{dk}_{\text{tag-f},j})_j, \text{tag-f}, F); (\text{ct}_{\text{tag},i})_i \leftarrow (\text{Enc}(\text{ek}_i, \text{tag}, x_i))_i \end{aligned}$$

where  $i \in [n]$  and  $j \in [m]$ , the following holds with overwhelming probability:

$$\text{Dec}(\text{dk}_{\text{tag-f},F}, (\text{ct}_{\text{tag},i})_{i=1}^n) = F(x_1, \dots, x_n) \text{ when } F(x_1, \dots, x_n) \neq \perp^7 \quad (3)$$

where  $F : \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n} \rightarrow \mathcal{R}_\lambda$  and the probability is taken over the random coins of algorithms.

**Security.** We follow the approach in the work by Chotard *et al.* [CDG<sup>+</sup>18a] so as to define the security game with oracles **Initialize**, **Corrupt**, **LoR**, **Enc**, **DKeyGenShare**, and **Finalize**. We recall that the oracle **Enc** is necessary for a simpler notion of one challenge, while retaining an equivalence to the multi-challenge notion using a hybrid argument shown in Lemma 8. The adversary is also able to corrupt *separately* the secret key  $\text{sk}_j$  of any key-generator  $j$  as well as the encryption key  $\text{ek}_i$  of any client  $i$ , which is done via requests  $(i, \text{skey})$  or  $(j, \text{ekey})$  to the oracle **Corrupt**, respectively. We need to exclude trivial attacks that can be mounted in the security experiment. Those restrictions are encompassed in the notion of *admissibility*, which is extended from similar notions in the works of [CDG<sup>+</sup>18a, CDSG<sup>+</sup>20].

In a nutshell, Definition 6 gives the definition of admissibility, generalizing the admissibility condition that has been consistently used since the seminal work of Chotard *et al.* [CDG<sup>+</sup>18a]. We refer to Section 3 for a high-level discussion. In the subsequent Section 4.1, we give the full formal treatment to prove the *optimality* of our admissibility condition in Definition 6. The successive security analyses of our DMCFE schemes rely crucially on this definition, translated for the concrete class of inner product in Remark 15.

**Definition 6 (Admissibility condition).** Let  $\mathcal{A}$  be a ppt adversary and let

$$\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$$

be a DMCFE scheme for a function class  $\mathcal{F}$  set up w.r.t  $\lambda \in \mathbb{N}$ . In **Finalize**, considering the queries  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(b)})\})$ , we say that the attack corresponding to these queries is NOT admissible if the following is satisfied

There exist  $\text{tag}, \text{tag-f} \in \text{Tag}$ , a function  $F \in \mathcal{F}$  having parameters  $\mathbf{y} \in \text{Param}_1 \times \cdots \times \text{Param}_m$ , two challenges  $(x_i^{(0)}, x_i^{(1)})_{i \in [n]}$  such that  $(\text{tag-f}, F) \in \mathcal{Q}$  is queried to **DKeyGenShare**,  $((x_i^{(0)}, x_i^{(1)})_{i \in [n]}, \text{tag})$  is queried to **LoR** and there exists a pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$ , a function  $G$  deducible from  $\mathbf{y}_{\text{skey}}$  satisfying

$$G(\mathbf{z}^{(0)}) \neq G(\mathbf{z}^{(1)}) , \quad (4)$$

where we define  $\mathbf{y}_{\text{skey}} := (\mathbf{y}[i])_{i \in \mathcal{H}_{\text{skey}}}$  and for  $b \in \{0, 1\}$ ,  $\mathbf{x}_{\text{ekey}}^{(b)} := (x_i^{(b)})_{i \in \mathcal{H}_{\text{ekey}}}$ .

Otherwise, we say that the aforementioned attack is admissible.

**Definition 7 (IND-security for DMCFE).** A DMCFE scheme

$$\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$$

<sup>7</sup> See [BO13, ABN10] for discussions about this relaxation. The general reason is that some functionality might contain  $\perp$  in its range and if  $F((x_i)_i) = \perp$  we do not impose  $\text{Dec}(\text{dk}_{\text{tag-f},F}, (\text{Enc}(\text{ek}_i, x_i, \text{tag}))_{i=1}^n) = F((x_i)_i)$ , neither do we disallow it.

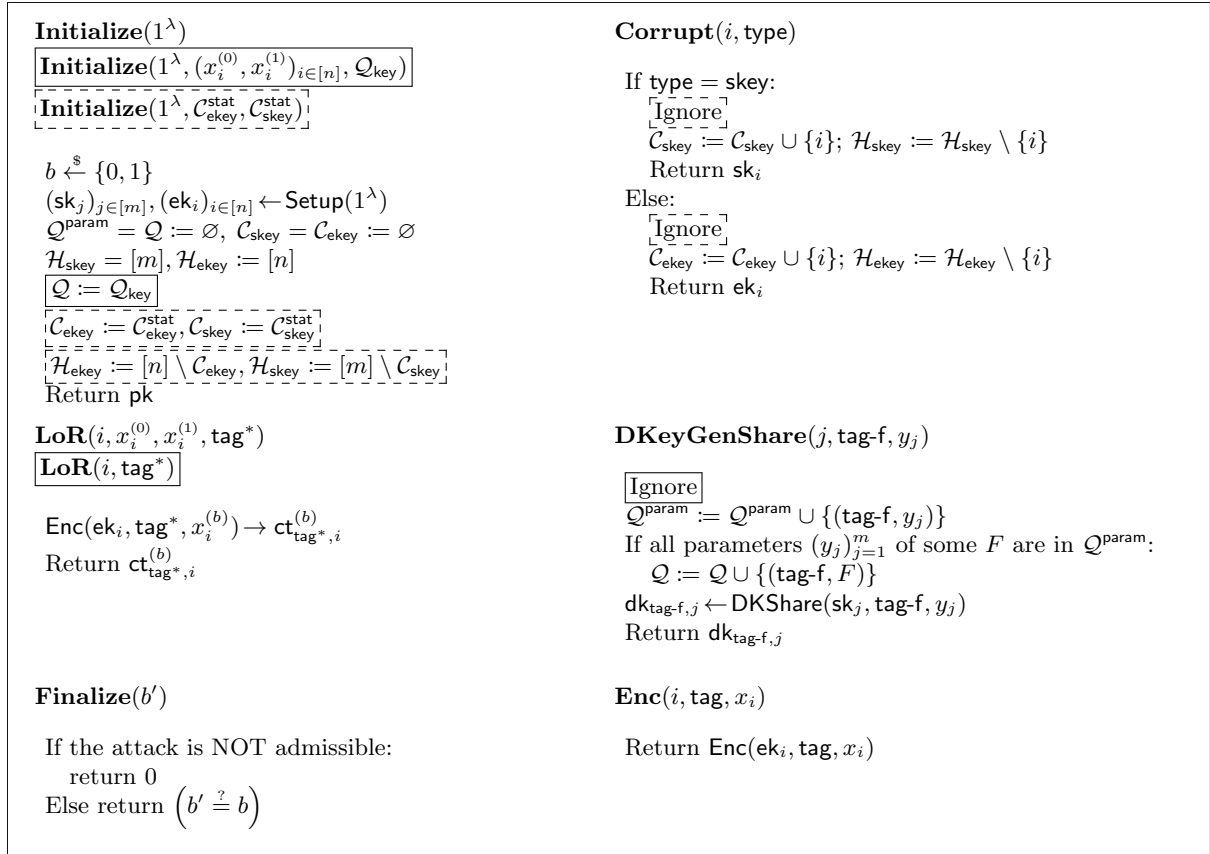


Fig. 1: The security games  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda)$ ,  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-sel-ind-cpa}}(1^\lambda)$ , and  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-stat-ind-cpa}}(1^\lambda)$  for Definition 7 and 17. The admissibility condition is defined in Definition 6.

for the function class  $\mathcal{F} = \{F_\lambda\}_{\lambda \in \mathbb{N}}$  is  $\text{xx}$ -secure if for all ppt adversaries  $\mathcal{A}$ , and for all sufficiently large  $\lambda \in \mathbb{N}$ , the following probability is negligible

$$\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}}(1^\lambda) := \left| \Pr[\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}}(1^\lambda) = 1] - \frac{1}{2} \right|.$$

The game  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}}(1^\lambda)$  is depicted in Figure 1. The security level indicator  $\text{xx}$  can be: **dmc-ind-cpa** to indicate IND-security with adaptive challenges, dynamic corruption of ekey, and dynamic corruption of skey; **dmc-sel-ind-cpa** to indicate selective IND-security with selective challenges, dynamic corruption of ekey, and dynamic corruption of skey; **dmc-stat-ind-cpa** to indicate static IND-security with adaptive challenges, static corruption of ekey, and static corruption of skey<sup>8</sup>; **dmc-ind-cpa-1chal** indicate one-time IND-security with one adaptive challenge tag, dynamic corruption of ekey, and dynamic corruption of skey. The probability is taken over the random coins of  $\mathcal{A}$  and the algorithms.

Lemma 8 allows us to concentrate on the notion of one-time IND-security for our DMCFE constructions. The proof is a standard hybrid argument and we give it in the appendix C.1 for completeness.

**Lemma 8.** *Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$  be a DMCFE scheme for the function class  $\mathcal{F}$ . If  $\mathcal{E}$  is one-time IND-secure, then  $\mathcal{E}$  is IND-secure.*

<sup>8</sup> In addition, we can allow dynamic corruption on one type but static corruption on the other type of keys, such as **dmc-stat-sk-ind-cpa** to indicate *partially static IND-security* with adaptive challenges, dynamic corruption of ekey, and static corruption of skey.

#### 4.1 Optimality of Admissibility *as per* Definition 6

In this section, we demonstrate that our notion of admissibility in Definition 6 is capturing all trivial attacks against DMCFE schemes for non-trivial function classes, which include the class of inner-product and quadratic functionalities. The high-level plan is given below.

**PROVING OPTIMALITY.** The general idea on defining the *optimality* of an admissibility condition can be revisited in Section 3.3. We now explain briefly how one can show that an admissibility condition is optimal following what we try to define. First and foremost, a notion of optimality makes sense when we consider only certain functionalities and not any arbitrary class of functions. For example, for a general functionality, the adversary's admissibility as defined in Definition 6 might not be *efficiently* decidable. Roughly speaking, **Finalize** may have to go through *all*  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$  and *all*  $G$  deducible from  $\mathbf{y}_{\text{skey}}$  so as to check relation (4). Therefore, we want to focus on classes for which the admissibility can be decided efficiently by **Finalize**, at least for the sake of having an efficient challenger in the security game.

In addition, we require a further property of the functionality under consideration: in view of the admissibility check (4), the deduction of  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$  and of a function  $G$  from  $\mathbf{y}_{\text{skey}}$  can be done efficiently. We coin this property *fixed-component distinguishability*. In summary, we restrain the optimality evaluation to classes that are (1) fixed-component distinguishable and (2) for which the admissibility is efficiently decidable. In the appendix B.1, we prove that both most studied functionalities for inner products and quadratic evaluations satisfy properties (1) and (2).

The core of our reasoning that an admissibility condition is optimal comprises building a ppt distinguisher, which can exert a *non-admissible* attack, to trivially win significantly the security game against *any* DMCFE scheme. We recall that Definition 6 views attacks as ensembles of queries made by some adversary in its security game. Because the class allows deciding efficiently the admissibility, our distinguisher can efficiently determine which query in the attack will violate the check (4), and thanks to the fixed-component distinguishability, the triplet  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, G)$  can be concretely reconstructed in an efficient manner.

In the end, facing any DMCFE challenger that allows the foregoing non-admissible behaviour, our distinguisher can simply use  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)}, G)$  to trivially win the game. This means that whenever we allow a non-admissible attack, or in other words whenever we try to relax Definition 6, no DMCFE scheme can be proved secure due to the existence of the above distinguisher.

To begin our formal treatment, we restrain our attention to particular function classes that satisfy certain properties.

**Definition 9 (Fixed-component distinguishable classes).** Fix  $\lambda \in \mathbb{N}$  and denote by  $\mathcal{F}_\lambda = \{F : \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n} \rightarrow \mathcal{R}_\lambda\}$  a family of  $n$ -ary functions indexed by  $\lambda$  having  $m$  parameters, where  $m = m(\lambda), n = n(\lambda)$  are functions.

For  $F \in \mathcal{F}_\lambda$ , a triple  $(\mathbf{x}_{\text{inp}}^{(0)}, \mathbf{x}_{\text{inp}}^{(1)}, \mathcal{H}_{\text{inp}})$ , where  $\mathbf{x}_{\text{inp}}^{(b)} \in \prod_{i \in \mathcal{H}_{\text{inp}}} \mathcal{D}_{\lambda,i}$  for  $b \in \{0, 1\}$ , is said to be distinguishing  $F_\lambda$  with fixed components if there exists a deducible pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)}) \in \prod_{i \in [n]} \mathcal{D}_{\lambda,i}$  such that  $F_\lambda(\mathbf{z}^{(0)}) \neq F_\lambda(\mathbf{z}^{(1)})$  where

$$\begin{cases} \mathbf{z}^{(b)}[i] = \mathbf{x}_{\text{inp}}^{(b)}[i] \text{ for } b \in \{0, 1\}, i \in \mathcal{H}_{\text{inp}} \\ \mathbf{z}^{(0)}[i] = \mathbf{z}^{(1)}[i] \forall i \in [n] \setminus \mathcal{H}_{\text{inp}} \end{cases}.$$

A function  $F$  is said to be fixed-component distinguishable if there exists a triple distinguishing  $F$  with fixed components and a ppt Turing machine that, given as input this triple, outputs the corresponding deducible pair.

A function class  $\mathcal{F}_\lambda$  is fixed-component distinguishable if for all  $F \in \mathcal{F}_\lambda$  with parameters  $\mathbf{p} := \mathbf{p}(F)$ , there exists a fixed-component distinguishable function  $G$  deducible from  $(\mathbf{p}[i])_{i \in \mathcal{H}_{\text{func}}}$  for some  $\mathcal{H}_{\text{func}} \subseteq [m]$ , and a ppt Turing machine that, given as inputs  $(F, \mathcal{H}_{\text{func}})$ , outputs  $G$ .

**Remark 10.** We remark that a function class  $\mathcal{F}$  is fixed-component distinguishable does *not* necessarily imply that the admissibility from Definition 6 for  $\mathcal{F}$  can be efficiently decided. Roughly speaking, given a function among the adversary’s queries, the ppt Turing machine from fixed-component distinguishability will output *some* deducible function  $G$  for which one can test the admissible condition (4) efficiently. But that is not enough, as to decide the admissibility of an attack, we need to check *all* such deducible functions and there is no further guarantee in the case of general functionalities that we can do this check efficiently. In the concrete cases of inner products and quadratic functions, the check over all such deducible functions can be done efficiently by using their linear properties, see the appendix B.1 for more details.

We now define what means for an admissibility to be *optimal* for a function class  $\mathcal{F}$ . For simplicity, we can consider the one-challenge setting thanks to Lemma 8.

**Definition 11.** Let  $\lambda \in \mathbb{N}$  and denote by  $\mathcal{F}$  a family of  $n(\lambda)$ -ary functions indexed by  $\lambda$ , with  $m(\lambda)$  parameters for each member of  $\mathcal{F}$ . An admissibility condition  $\text{adm}(\cdot)$  is optimal for  $\mathcal{F}$  if there exists a ppt distinguisher  $\mathcal{S}$  so that for all non-admissible

$$(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$$

of some ppt adversary  $\mathcal{A}$  and against a DMCFE  $\mathcal{E}$  for  $\mathcal{F}$  in a security experiment  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}$  given in Figure 1, we have

$$\Pr [\mathcal{S}((\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})) = b : b \leftarrow \text{Chall}(\text{rand}_{\text{Chall}})] \geq \frac{1}{\text{poly}(\lambda)}$$

where  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$  is well-defined at the time of **Finalize** and  $b \leftarrow \text{Chall}(\text{rand}_{\text{Chall}})$  means the challenger  $\text{Chall}$  uses the bit  $b$  in  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}$ .

We now state our main theorem for the optimality of our admissibility.

**Theorem 12.** Let  $\mathcal{F}$  be a function class that has efficient decidability for admissibility and is fixed-component distinguishable. Then, our admissibility condition as defined in Definition 6 is optimal for  $\mathcal{F}$ .

*Proof (Of Theorem 12).* Without loss of generality, we consider the one-challenge notion. We need to prove that: there exists a ppt distinguisher  $\mathcal{S}$  so that for any non-admissible  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$  of some DMCFE  $\mathcal{E}$  for  $\mathcal{F}$  and some ppt adversary  $\mathcal{A}$  in a security experiment  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}}$  given in Figure 1, we have

$$\Pr [\mathcal{S}((\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})) = b : b \leftarrow \text{Chall}(\text{rand}_{\text{Chall}})] \geq \frac{1}{\text{poly}(\lambda)} .$$

Let  $\mathcal{E}^{\text{abs}} = (\text{Setup}^{\text{abs}}, \text{DKShare}^{\text{abs}}, \text{DKeyComb}^{\text{abs}}, \text{Enc}^{\text{abs}}, \text{Dec}^{\text{abs}})$  be an abstract DMCFE for  $\mathcal{F}$  that satisfies the correctness relation (3). We describe the distinguisher  $\mathcal{S}$  as follows:

1. The distinguisher  $\mathcal{S}$  parses

$$(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$$

then use  $\mathcal{E}^{\text{abs}}$  and  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\})$  for abstracting the key components to obtain  $\{(\text{dk}_{\text{tag-f}, F, j}^{\text{abs}})_{j \in [m]}\}_{(\text{tag-f}, F) \in \mathcal{Q}}$ , the challenge ciphertext components to obtain  $(\text{ct}_{\text{tag}, i}^{\text{abs}})_{i \in [n]}$  for each  $\{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}$ , and the encryption responses to obtain  $(\text{ct}_i^{\text{abs}, (k)})_{i \in [n]}$ . If there are corrupted keys  $\text{sk}_j$  or  $\text{ek}_i$  queried by  $\mathcal{A}$ , they will also be replaced by their abstracted counterparts  $\text{sk}_j^{\text{abs}}$  or  $\text{ek}_i^{\text{abs}}$ . In the following  $\mathcal{S}$  only needs the abstract DMCFE  $\mathcal{E}^{\text{abs}}$  for  $\mathcal{F}$ , no matter what the details of the concrete  $\mathcal{E}$  are.

2. If there exists  $(\text{tag-f}, F) \in \mathcal{Q}$  such that  $F(\mathbf{x}_0^*) \neq F(\mathbf{x}_1^*)$ ,  $\mathcal{S}$  combines the key components of  $(\text{tag-f}, F)$ , decrypts the challenge ciphertext components, and outputs 1 if and only if the result is  $F(\mathbf{x}_1^*)$ . All algorithms come from  $\mathcal{E}^{\text{abs}} = (\text{Setup}^{\text{abs}}, \text{DKShare}^{\text{abs}}, \text{DKeyComb}^{\text{abs}}, \text{Enc}^{\text{abs}}, \text{Dec}^{\text{abs}})$ . Else, in the following we assume that  $F(\mathbf{x}_0^*) = F(\mathbf{x}_1^*)$  for all  $(\text{tag-f}, F) \in \mathcal{Q}$ .
3. Because this is a non-admissible attack,  $\mathcal{S}$  uses the efficient decidability of  $\mathcal{F}$  to find  $(\text{tag-f}, F) \in \mathcal{Q}$ , whose parameters is  $\mathbf{y} := \mathbf{p}(F)$ , so that: there exists a pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$ , a function  $G$  deducible from  $\mathbf{y}_{\text{skey}}$  satisfying

$$G(\mathbf{z}^{(0)}) \neq G(\mathbf{z}^{(1)}) ,$$

where  $\mathbf{y}_{\text{skey}} := (\mathbf{y}[i])_{i \in \mathcal{H}_{\text{skey}}}$  and  $\mathbf{x}_{\text{ekey}}^{(0)} := (x_i^{(0)})_{i \in \mathcal{H}_{\text{ekey}}}$ ,  $\mathbf{x}_{\text{ekey}}^{(1)} := (x_i^{(1)})_{i \in \mathcal{H}_{\text{ekey}}}$ . We remark that finding  $F$  can be done efficiently in  $\mathcal{Q}$  because the current attack comes from the execution of some ppt adversary  $\mathcal{A}$ , which implies the size of  $\mathcal{Q}$  is polynomially bounded.

4. Because  $\mathcal{F}$  is fixed-component distinguishable (see Definition 9), using  $F$  and  $\mathcal{C}_{\text{skey}}$ ,  $\mathcal{S}$  can efficiently find a function  $G$  deducible from  $\mathbf{y}_{\text{skey}}$  such that  $G$  is fixed-component distinguishable.
5. Thanks to the fixed-component distinguishability of  $G$ , using  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$  and  $\mathcal{H}_{\text{ekey}}$ , the pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  can be found efficiently by  $\mathcal{S}$ .
6. The distinguisher  $\mathcal{S}$  then uses the corrupted keys  $(\text{ek}_i^{\text{abs}})_{i \in \mathcal{C}_{\text{ekey}}}$  to compute new ciphertext components  $(\tilde{\text{ct}}_{\text{tag},i}^{\text{abs}})_{i=1}^n$  of  $\mathbf{z}^{(b)}$  implicitly, using  $(\text{ct}_{\text{tag},i}^{\text{abs}})_{i \in \mathcal{H}_{\text{ekey}}}$  for the challenge  $(\mathbf{x}_b^*[i])_{i \in \mathcal{H}_{\text{ekey}}}$ , and using  $\text{Enc}^{\text{abs}}$  to encrypt  $(\mathbf{z}^{(b)}[i])_{i \in \mathcal{C}_{\text{ekey}}}$  using  $(\text{ek}_i^{\text{abs}})_{i \in \mathcal{C}_{\text{ekey}}}$ .
7. Next,  $\mathcal{S}$  uses the corrupted keys  $(\text{sk}_i^{\text{abs}})_{i \in \mathcal{C}_{\text{skey}}}$  to compute new key components  $(\tilde{\text{dk}}_{\text{tag-f},G,i}^{\text{abs}})_{i=1}^n$  of  $G$  implicitly, using  $(\text{dk}_{\text{tag-f},F,i}^{\text{abs}})_{i \in \mathcal{H}_{\text{skey}}}$ , and using  $\text{DKShare}^{\text{abs}}$  to derive  $(\tilde{\text{dk}}_{\text{tag-f},F,i}^{\text{abs}})_{i \in \mathcal{C}_{\text{skey}}}$  from  $(\mathbf{p}(G)[i])_{i \in \mathcal{C}_{\text{skey}}}$ .
8. Finally,  $\mathcal{S}$  uses  $\text{DKeyComb}^{\text{abs}}$  to combine the newly generated key components  $(\tilde{\text{dk}}_{\text{tag-f},i}^{\text{abs}})_{i=1}^n$ . Then  $\mathcal{S}$  decrypts the newly generated challenge ciphertext  $(\tilde{\text{ct}}_{\text{tag},i}^{\text{abs}})_{i \in [n]}$  using the abstract algorithm  $\text{Dec}^{\text{abs}}$ , the adversary outputs 1 if and only if the result is equal to  $G(\mathbf{z}^{(1)})$ .

In the end,  $\mathcal{S}$  outputs 1 if and only if  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$  comes from an execution of any  $\mathcal{A}$  against  $\text{Chall}$  of  $\mathcal{E}$  in which  $\text{Chall}$  picks 1 as the challenge bit. This concludes the proof.  $\square$

**Remark 13.** The abstract object  $\mathcal{E}^{\text{abs}}$  in the proof of Theorem 12 are used *only* in our formal proofs of the optimality for our admissible condition. In the concrete constructions of DMCFE, no such abstract objects are needed, as the admissibility are examined via concrete tests over the adversary's queries in the security game. For instance, see the appendix B.1 for the cases of linear and quadratic functions.

**Remark 14.** The generic distinguisher  $\mathcal{S}$  in Theorem 12 is *weak* in the sense that all it has is a non-admissible attack with the corresponding

$$(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$$

determined by  $\mathcal{A}$ 's behaviour during the security game, *not depending on* the concrete implementation of  $\mathcal{E}$ . However, thanks to the non-admissibility of the given attack and the fixed-component distinguishability of the function class,  $\mathcal{S}$  can still output the correct challenge bit, in the security against *any* DMCFE scheme. This means that as soon as we allow some non-admissible behaviour, where the concrete descriptions of  $\mathcal{A}$  and  $\mathcal{E}$  can be arbitrary as long as this behaviour stays the same, there is no hope in proving security regardless of the specific implementation of  $\mathcal{E}$ . Equivalently, our Definition 6 that excludes exactly these non-admissible attacks cannot be enlarged in any sense and captures the most attacks against which we can prove security. Last but not least, we see clearly the role of the abstract DMCFE  $\mathcal{E}^{\text{abs}}$ : it abstracts out the concrete details of *some* specific scheme  $\mathcal{E}$  from which calculations over the non-admissible queries can be done, and return the “black-boxed” data that obey the correctness of DMCFE schemes for  $\mathcal{F}$ .

**Remark 15.** As a corollary the admissibility's optimality for the class of inner products (including for polynomially bounded ranges - see the appendix B.1 for more details), we have specific conditions for admissible attacks in this case:

1. For all vectors  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  that is queried to **LoR**, for all  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$ ,  $\sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] = 0$  where  $\Delta \mathbf{x}[i] = \mathbf{x}_0^*[i] - \mathbf{x}_1^*[i]$ , where  $\mathcal{H} := \mathcal{H}_{\text{ekey}} \cap \mathcal{H}_{\text{skey}}$ .
2. For all vectors  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  that is queried to **LoR**, for all  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$ , for all  $i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ , either  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i] = 0$  or  $\mathbf{y}[i] = 0$ .
3. For all vectors  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  that is queried to **LoR**, for all  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$ , for all  $i \in \mathcal{C}_{\text{skey}}$ ,  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$ .

## 5 DMCFE for Inner Products with Stronger Security against Complete Queries

### 5.1 Construction

This section presents a *decentralized* multi-client FE scheme, as defined in Section 4, for the function class  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$  and  $F_{\mathbf{y}} : (\mathbb{Z}_q^*)^n \rightarrow \mathbb{Z}_q$  is defined as  $F_{\mathbf{y}}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle$  where  $\|\mathbf{x}\|_{\infty} < B$  and  $\|\mathbf{y}\|_{\infty} < B'$ , where  $B, B' = \text{poly}(\lambda) \in \mathbb{N}$  are polynomials. The high-level ideas are discussed in Section 3.4. As discussed in Remark 4, the parameter vector of  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$  is simply  $\mathbf{y}$  of size  $n$ . Hence the number of secret keys and of encryption keys are equal to  $n$ . Our admissibility is also optimal for  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$ , see the appendix B.1. We need a full-domain hash function  $\mathbf{H}_1 : \text{Tag} \rightarrow \mathbb{G}_1^2$ , where  $\text{Tag}$  denotes the set of tags used for ciphertext components and functional key components. In addition, we also need a hash function  $\mathbf{H}_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ .

We are in the bilinear group setting  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$  and  $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$  are written additively. The details of our DMCFE scheme go as follows:

**Setup**( $1^\lambda$ ): Choose  $n$  pairs of dual orthogonal bases  $(\mathbf{H}_i, \mathbf{H}_i^*)$  for  $i \in [n]$ , where  $(\mathbf{H}_i, \mathbf{H}_i^*)$  is a pair of dual bases for  $(\mathbb{G}_1^6, \mathbb{G}_2^6)$ . We denote the basis changing matrices for  $(\mathbf{H}_i, \mathbf{H}_i^*)$  as  $(H_i, H_i')$ :

$$(\mathbf{H}_i = H_i \cdot \mathbf{T}; \mathbf{H}_i^* = H_i' \cdot \mathbf{T}^*)_{i \in [n]}$$

where  $H_i, H_i' \in \mathbb{Z}_q^{6 \times 6}$  and  $(\mathbf{T} = \llbracket I_6 \rrbracket_1, \mathbf{T}^* = \llbracket I_6 \rrbracket_2)$  are canonical bases of  $(\mathbb{G}_1^6, \mathbb{G}_2^6)$ , for the identity matrix  $I_6$ . Sample two full-domain hash functions  $\mathbf{H}_1 : \text{Tag} \rightarrow \mathbb{G}_1^2$  and  $\mathbf{H}_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ . We recall that interactions are involved only in this **Setup** phase. For each  $i \in [n]$ , we write

$$\mathbf{H}_i = (\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \dots, \mathbf{h}_{i,6}) \quad \mathbf{H}_i^* = (\mathbf{h}_{i,1}^*, \mathbf{h}_{i,2}^*, \dots, \mathbf{h}_{i,6}^*)$$

and sample  $S, U, V, T \stackrel{\$}{\leftarrow} (\mathbb{Z}_q^*)^n$  where  $S = (s_1, \dots, s_n)$ ,  $U = (u_1, \dots, u_n)$ ,  $V = (v_1, \dots, v_n)$ ,  $T = (t_1, \dots, t_n)$ . Sample  $\theta_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q^*$  such that  $\sum_{i=1}^n \theta_i = 0$ . Then, sample  $\zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma_i' \stackrel{\$}{\leftarrow} \mathbb{Z}_q$ , for  $i \in [n]$ , satisfying

$$p_i \alpha_i = \zeta_1 \quad q_i \gamma_i = \zeta_2 \quad q_i \alpha_i = \zeta_3 \quad p_i \gamma_i' = \zeta_4 \quad (5)$$

and output the *secret keys*  $\text{sk}_i$  and the *encryption keys*  $\text{ek}_i$  as follows

$$\begin{aligned} \text{sk}_i &:= \left( s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*, u_i \gamma_i' \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*, \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*, \theta_i \mathbf{h}_{i,6}^* \right) \\ \text{ek}_i &:= (p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)}, q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)}) \end{aligned}$$

where  $H_i^{(k)}$  denotes the  $k$ -th row of  $H_i$  for  $i \in [n]$ .



**DKShare**( $\text{sk}_i, (\text{tag-f}, \text{info}(\mathbf{y})), y_i$ ): We assume that the function  $\text{tag}$  contains  $\text{tag-f}$  and public information about  $\text{info}(\mathbf{y})$ . The  $i$ -th parameter is  $y_i := \mathbf{y}[i]$ . Compute  $H_2(\text{tag-f}, \text{info}(\mathbf{y})) \rightarrow d_{\text{tag-f}, \mathbf{y}} \in \mathbb{Z}_q$ . Parse

$$\text{sk}_i := \left( s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*, u_i \gamma_i' \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*, \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*, \theta_i \mathbf{h}_{i,6}^* \right).$$

Sample  $z \xleftarrow{\$} \mathbb{Z}_q$  then compute

$$\begin{aligned} \mathbf{k}_{i,\text{ipfe}} &= y_i \cdot (s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*) + y_i \cdot (u_i \gamma_i' \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*) + y_i \cdot \left( \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^* \right) + d_{\text{tag-f}, \mathbf{y}} \cdot \theta_i \mathbf{h}_{i,6}^* \\ &= (s_i y_i \alpha_i + u_i y_i \gamma_i', s_i y_i \gamma_i + u_i y_i \alpha_i, \frac{v_i}{t_i} y_i, y_i, 0, d_{\text{tag-f}, \mathbf{y}} \theta_i)_{\mathbf{H}_i^*}. \end{aligned}$$

Output  $\text{dk}_{\text{tag-f}, i} := \mathbf{k}_{i,\text{ipfe}}$ .

**DKeyComb**(( $\text{dk}_{\text{tag-f}, i}$ ) $_{i \in [n]}$ ,  $\text{tag-f}, \mathbf{y}$ ): Output  $\perp$  if there is any incoherence of  $d_{\text{tag-f}, \mathbf{y}}$  among the  $\text{dk}_{\text{tag-f}, i}$ . Otherwise, parse  $\text{dk}_{\text{tag-f}, i} := \mathbf{k}_{i,\text{ipfe}}$  and output  $\text{dk}_{\text{tag-f}, \mathbf{y}} := (\mathbf{k}_{i,\text{ipfe}})_{i \in [n]}$ .

**Enc**( $\text{ek}_i, \text{tag}, x_i$ ): Parse

$$\text{ek}_i := (p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)}, q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)})$$

and compute  $H_1(\text{tag}) \rightarrow ([\omega]_1, [\omega']_1) \in \mathbb{G}_1^2$  and sample  $r_i \xleftarrow{\$} \mathbb{Z}_q$ . Compute

$$\begin{aligned} \mathbf{c}_{i,\text{ipfe}} &= (p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)}) \cdot [\omega]_1 + (q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}) \cdot [\omega']_1 \\ &\quad + r_i \cdot (t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}) + x_i \mathbf{h}_{i,4} + H_i^{(6)} [\omega]_1 \\ &= (\omega p_i, \omega' q_i, r_i t_i, -(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i - r_i v_i, 0, \omega)_{\mathbf{H}_i}. \end{aligned}$$

and output  $\text{ct}_{\text{tag}, i} := \mathbf{c}_{i,\text{ipfe}}$ .

**Dec**( $\text{dk}_{\text{tag-f}, \mathbf{y}}, \mathbf{c}$ ): Parse  $\text{dk}_{\text{tag-f}, \mathbf{y}} = (\mathbf{k}_{i,\text{ipfe}})_{i \in [n]}$  and  $\mathbf{c} := (\text{ct}_{\text{tag}, i})_i$ . Finally, compute the discrete logarithm in base  $g_t$  of  $[\text{out}]_t = \sum_{i=1}^n (\text{ct}_{\text{tag}, i} \times \mathbf{k}_{i,\text{ipfe}})$  and output the small value  $\text{out}$ .

The *correctness* of the scheme is verified by:

$$\begin{aligned} & [\text{out}]_t \\ &= \sum_{i=1}^n (\text{ct}_{\text{tag}, i} \times \mathbf{k}_{i,\text{ipfe}}) \\ &= \sum_{i=1}^n \left( \begin{array}{c} (\omega p_i, \omega' q_i, r_i t_i, -(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i - r_i v_i, \\ 0, \theta_i)_{\mathbf{H}_i} \\ \times \\ (s_i y_i \alpha_i + u_i y_i \gamma_i', s_i y_i \gamma_i + u_i y_i \alpha_i, -\frac{v_i}{t_i} y_i, y_i, \\ 0, d_{\text{tag-f}, \mathbf{y}})_{\mathbf{H}_i^*} \end{array} \right) \\ &\stackrel{(*)}{=} \sum_{i=1}^n [\omega \zeta_1 s_i y_i + \omega \zeta_4 u_i y_i + \omega' \zeta_2 s_i y_i + \omega' \zeta_3 u_i y_i + \theta_i d_{\text{tag-f}}]_t \\ &\quad + \sum_{i=1}^n [(-(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i) \cdot y_i]_t \\ &= [(\omega \zeta_1 + \omega' \zeta_2) \cdot \langle S, \mathbf{y} \rangle + (\omega' \zeta_3 + \omega \zeta_4) \cdot \langle U, \mathbf{y} \rangle]_t + \sum_{i=1}^n [\theta_i d_{\text{tag-f}}]_t \\ &\quad - [(\omega \zeta_1 + \omega' \zeta_2) \cdot \langle S, \mathbf{y} \rangle + (\omega' \zeta_3 + \omega \zeta_4) \cdot \langle U, \mathbf{y} \rangle]_t + [\langle \mathbf{x}, \mathbf{y} \rangle]_t \\ &= [\langle \mathbf{x}, \mathbf{y} \rangle]_t, \end{aligned}$$

where the equality (\*) comes from system (5). We recall that  $(\theta_i)_{i \in [n]}$  is a secret sharing of 0.

## 5.2 Adaptive Security against Static Corruptions of Secret Keys

We now give the security theorem of one time IND-security for our construction from Section 5.1, *adaptively* in the challenge messages with *dynamic* corruption of encryption keys and *static* corruption of secret keys. We refer to Remark 15 for the concrete interpretation of the security model. The full proof can be found in the appendix C.2.

**Theorem 16.** *Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$  be a DMCFE candidate for  $\mathcal{F}^{\text{IP}}$  from Section 5.1 in a bilinear group  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ . Then,  $\mathcal{E}$  is IND-secure with static corruption of secret keys in the ROM if the SXDH assumption holds for  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . More specifically, let  $n$  denote the dimension for inner-products,  $K$  denote the maximum number of key queries, and  $Q_1, Q_2$  denote the maximum number of random oracle (RO) queries to  $\mathbf{H}_1, \mathbf{H}_2$  respectively. For any ppt adversary  $\mathcal{A}$  against  $\mathcal{E}$  with static secret key corruption and one-time challenge, we have the following bound:*

$$\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-stat-sk-ind-cpa-1chal}}(1^\lambda) \leq (3 + 2Q_1 + K) \cdot \text{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\text{SXDH}}(1^\lambda) + \frac{Q_2^2}{2q}$$

and in the reduction there is an additive loss  $\mathcal{O}(Q_1 \cdot t_{\mathbb{G}_1} + Q_2 \cdot t_{\mathbb{G}_2})$  in time, where  $t_{\mathbb{G}_1}, t_{\mathbb{G}_2}$  is the cost for one addition in  $\mathbb{G}_1, \mathbb{G}_2$ .

## 6 DMCFE for Inner-Products with Stronger Security against Incomplete Queries

In this section, we show how to obtain a DMCFE scheme that is IND-secure against chosen plaintext attacks without *complete* queries restriction (see condition 2 in Section 3.1), under our stronger admissibility notion. The definition of security notion for the new setting is restated so that admissible adversaries can query *incomplete* challenge ciphertexts as well as *incomplete* functional keys.

**Definition 17 (Admissible attacks with incomplete queries).** *Let  $\mathcal{A}$  be a ppt adversary and let*

$$\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$$

be a DMCFE scheme for a function class  $\mathcal{F}$  set up w.r.t  $\lambda \in \mathbb{N}$ . We denote by  $\text{rand}_{\text{Chall}}$  the random coins of the challenger and  $\text{rand}_{\mathcal{A}}$  the random coins of the adversary in an experiment given in Figure 1. In **Finalize**, considering the queries  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$ , we say that the attack corresponding to these queries is NOT admissible if the following is satisfied

There exist  $\text{tag}, \text{tag-f} \in \text{Tag}$ , a function  $F \in \mathcal{F}$ , two challenges  $(x_i^{(0)}, x_i^{(1)})_{i \in [n]}$  such that  $(F, \text{tag-f})$  is queried to **DKeyGenShare** for all honest components,  $((x_i^{(0)}, x_i^{(1)})_{i \in [n]}, \text{tag})$  is queried to **LoR** for all honest components, and there exists a pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$ , a function  $G$  deducible from  $\mathbf{y}_{\text{skey}}$  satisfying

$$G(\mathbf{z}^{(0)}) \neq G(\mathbf{z}^{(1)}) ,$$

where we define  $\mathbf{y}_{\text{skey}} := (y_i)_{i \in \mathcal{H}_{\text{skey}}}$  and for  $b \in \{0, 1\}$ ,  $\mathbf{x}_{\text{ekey}}^{(b)} := (x_i^{(b)})_{i \in \mathcal{H}_{\text{ekey}}}$ .

Otherwise, we say that the attack is admissible.

**Definition 18 (IND+-security for DMCFE).** *A DMCFE scheme*

$$\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$$

for the function class  $\mathcal{F} = \{F_\lambda\}_{\lambda \in \mathbb{N}}$  is IND+-secure if for all ppt adversaries  $\mathcal{A}$ , and for all sufficiently large  $\lambda \in \mathbb{N}$ , the following probability is negligible

$$\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}+}(1^\lambda) := \left| \Pr[\text{Exp}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}+}(1^\lambda) = 1] - \frac{1}{2} \right|.$$

The probability is taken over coins of  $\mathcal{A}$  and the algorithms. The indicator xx can be among  $\{\text{dmc-ind-cpa}, \text{dmc-sel-ind-cpa}, \text{dmc-stat-ind-cpa}, \text{dmc-ind-cpa-1chal}\}$ <sup>9</sup>. The experiment  $\text{Exp}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}+}(1^\lambda)$  is the same as  $\text{Exp}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{xx}}(1^\lambda)$  depicted in Figure 1, except that we use Definition 17 for the admissibility condition in **Finalize**.

## 6.1 Constructions

**Generic Transformation with Security against Selective Challenges.** We follow the same method in [CDSG<sup>+</sup>20] and apply generically a layer of using a primitive called *All-or-Nothing Encapsulation* (AoNE), so as to make our scheme from Section 5.1 secure in our *stronger* security model against incomplete queries. Our AoNE-based transformation uses the generic AoNE from [CDSG<sup>+</sup>20], which in turn is built on top of a *one-time secure symmetric encryption* (OT-SE). In the security proof, which can be naturally adapted from [CDSG<sup>+</sup>20, Theorem 26], this OT-SE prevents programming conveniently to achieve adaptive security w.r.t the challenge ciphertexts. We also remark that the static corruption is unavoidable since the security of AoNE makes sense only when being applied on honest components, for the security reduction. This generic transformation is provably secure under *static* corruption and *selective* challenges. The transformation is presented in the appendix B.2.

**Concrete Scheme with Security against Adaptive Challenges.** We present a concrete adaptation of our base DMCFE scheme from Section 5.1 to satisfy the stronger security notion against *incomplete* challenge ciphertexts as well as *incomplete* functional keys, with minimal modifications being put in boxed components for the ease of comparison. The function class stays the same as in Section 5.1, for which our admissibility is optimal, see the appendix B.1. In contrast to the generic transformation, we build the AoNE concretely by combining one-time pad (OTP) and a random oracle (RO). Then, the programmability of the RO helps us circumvent the problem of adaptive queries. While programming the RO, we indeed exploits in a non-blackbox manner the OTP as a summation in  $\mathbb{Z}_q^*$  to accumulate a secret sharing of 0 on the honest parts (known in advance thanks to static corruption).

The details of our construction go as follows:

**Setup( $1^\lambda$ ):** Sample two full-domain hash functions  $H_1 : \text{Tag} \rightarrow \mathbb{G}_1^2$  and  $H_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{G}_2$ . Choose  $n$  pairs of dual orthogonal bases  $(\mathbf{H}_i, \mathbf{H}_i^*)$  for  $i \in [n]$ , where  $(\mathbf{H}_i, \mathbf{H}_i^*)$  is a pair of dual bases for  $\boxed{(\mathbb{G}_1^8, \mathbb{G}_2^8)}$ . We denote the basis changing matrices for  $(\mathbf{H}, \mathbf{H}_i^*)$  as  $(H_i, H_i')$ :

$$(\mathbf{H}_i = H_i \cdot \mathbf{T}; \mathbf{H}_i^* = H_i' \cdot \mathbf{T}^*)_{i \in [n]}$$

where  $H_i, H_i' \in \mathbb{Z}_q^{8 \times 8}$  and  $(\mathbf{T} = [I_8]_1, \mathbf{T}^* = [I_8]_2)$  are canonical bases of  $(\mathbb{G}_1^8, \mathbb{G}_2^8)$ , for the identity matrix  $I_8$ . We recall that interactions are involved only in this Setup phase. For each  $i \in [n]$ , we write

$$\boxed{\mathbf{H}_i = (\mathbf{h}_{i,1}, \mathbf{h}_{i,2}, \dots, \mathbf{h}_{i,8}) \quad \mathbf{H}_i^* = (\mathbf{h}_{i,1}^*, \mathbf{h}_{i,2}^*, \dots, \mathbf{h}_{i,8}^*)}$$

<sup>9</sup> Similarly, we can allow dynamic corruption on one type but static corruption on the other type of keys, such as *dmc-stat-sk-ind-cpa+* to indicate *partially static IND-security* with adaptive challenges, dynamic corruption of ekey, and static corruption of skey.

and sample  $\zeta_1, \zeta_2, \zeta_3, \zeta_4, S, U, V, T, \overline{[D, E]} \xleftarrow{\$} (\mathbb{Z}_q^*)^n$  where  $S = (s_1, \dots, s_n)$ ,  $U = (u_1, \dots, u_n)$ ,  $V = (v_1, \dots, v_n)$ ,  $T = (t_1, \dots, t_n)$ ,  $D = (d_1, \dots, d_n)$ , and  $E = (e_1, \dots, e_n)$ . Sample  $\theta_1, \dots, \theta_n \xleftarrow{\$} \mathbb{Z}_q^*$  such that  $\sum_{i=1}^n \theta_i = 0$  and for  $i \in [n]$  let  $p_i, q_i, \alpha_i, \gamma_i, \gamma'_i \xleftarrow{\$} \mathbb{Z}_q$  satisfy

$$p_i \alpha_i = \zeta_1 \quad q_i \gamma_i = \zeta_2 \quad q_i \alpha_i = \zeta_3 \quad p_i \gamma'_i = \zeta_4$$

We set the public parameters to be  $\overline{[\langle E, \mathbf{1} \rangle]_1, \langle D, \mathbf{1} \rangle]_2}$ . Sample  $\overline{[\epsilon, \delta]} \xleftarrow{\$} \mathbb{Z}_q$  and generate random  $n$ -out-of- $n$  secret sharings  $\overline{[\epsilon_i]_i, [\delta_i]_i}$  of  $\epsilon, \delta$  so that  $\sum_{i=1}^n \epsilon_i = \epsilon$ ,  $\sum_{i=1}^n \delta_i = \delta$ . Output the *secret keys*  $\text{sk}_i$  and the *encryption keys*  $\text{ek}_i$  as follows

$$\begin{aligned} \text{sk}_i &:= (\overline{[\epsilon_i]}, s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*, u_i \gamma'_i \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*, \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*, \overline{[\theta_i H_i'^{(6)} - e_i H_i'^{(8)}, \mathbf{ch}_{i,8}]}) \\ \text{ek}_i &:= (\overline{[\delta_i]}, p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)} - \overline{[d_i H_i^{(7)}]}, q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, \\ &\quad H_i^{(6)}, \overline{[\delta \mathbf{h}_{i,7}^*]}) \end{aligned}$$

where  $H_i^{(k)}$  denotes the  $k$ -th row of  $H_i$  for  $i \in [n]$  and  $\mathbf{1} = (1, \dots, 1)$ .

**DKShare**( $\text{sk}_i, (\text{tag-f}, \text{info}(\mathbf{y})), y_i$ ): We assume that the function  $\text{tag}$  contains  $\text{tag-f}$  and public information about  $\text{info}(\mathbf{y})$ . The  $i$ -th parameter is  $y_i := \mathbf{y}[i]$ . We will use a full-domain hash function  $\text{H}_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{G}_2$ . Parse

$$\text{sk}_i := (\epsilon_i, s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*, u_i \gamma'_i \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*, \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*, \theta_i H_i'^{(6)} - e_i H_i'^{(8)}, \mathbf{ch}_{i,8}) .$$

Compute  $\text{H}_2(\text{tag-f}, \text{info}(\mathbf{y})) \rightarrow \overline{[\kappa_{\text{tag-f}, \mathbf{y}}]}_2$  and

$\mathbf{k}_{i, \text{ipfe}}$

$$\begin{aligned} &= y_i \cdot (s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*) + y_i \cdot (u_i \gamma'_i \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*) \\ &\quad + y_i (\frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*) + (\theta_i H_i'^{(6)} - e_i H_i'^{(8)}) \cdot \overline{[\kappa_{\text{tag-f}, \mathbf{y}}]}_2 \\ &= (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, \frac{v_i}{t_i} y_i, y_i, 0, \kappa_{\text{tag-f}, \mathbf{y}} \theta_i, \overline{[0, -e_i \kappa_{\text{tag-f}, \mathbf{y}}]})_{\mathbf{H}_i^*} \end{aligned}$$

$$\overline{\mathbf{e}(\epsilon_i \cdot \overline{[\langle E, \mathbf{1} \rangle]_1}, \overline{[\kappa_{\text{tag-f}, \mathbf{y}}]}_2)} = \overline{[\epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}}]}_{\mathbf{t}}$$

where  $\overline{[\langle E, \mathbf{1} \rangle]_1}$  is public. Output  $\text{dk}_{\text{tag-f}, i} := (\mathbf{k}_{i, \text{ipfe}}, \overline{[\epsilon \cdot \mathbf{h}_{i,8}, \overline{[\epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}}]}_{\mathbf{t}}]})$ .

**DKKeyComb**( $\text{dk}_{\text{tag-f}, i}, \text{tag-f}, \mathbf{y}$ ): Output  $\perp$  if there is any incoherence among the  $\text{dk}_{\text{tag-f}, i}$ . Else, let  $\text{dk}_{\text{tag-f}, i} := (\mathbf{k}_{i, \text{ipfe}}, \epsilon \cdot \mathbf{h}_{i,8}, \overline{[\epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}}]}_{\mathbf{t}})$ .

Compute  $\overline{[\epsilon \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}}]}_{\mathbf{t}} = \sum_{i=1}^n \overline{[\epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}}]}_{\mathbf{t}}$  and output

$$\text{dk}_{\text{tag-f}, \mathbf{y}} := ((\mathbf{k}_{i, \text{ipfe}}, \overline{[\epsilon \cdot \mathbf{h}_{i,8}]}_{i \in [n]}), \overline{[\epsilon \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}}]}_{\mathbf{t}}) .$$

**Enc**( $\text{ek}_i, \text{tag}, x_i$ ): Parse

$$\text{ek}_i := (\delta_i, p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)} - d_i H_i^{(7)}, q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)}, \delta \cdot \mathbf{h}_{i,7}^*)$$

and compute  $\text{H}_1(\text{tag}) \rightarrow (\overline{[\omega]}_1, \overline{[\omega']}_1) \in \mathbb{G}_1^2$ ; and sample  $r_i \xleftarrow{\$} \mathbb{Z}_q$ . Compute

$$\begin{aligned} \mathbf{c}_{i, \text{ipfe}} &= (p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)} - d_i H_i^{(7)}) \cdot \overline{[\omega]}_1 + (q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}) \cdot \overline{[\omega']}_1 \\ &\quad + r_i \cdot (t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}) + x_i \mathbf{h}_{i,4} + H_i^{(6)} \overline{[\omega]}_1 \\ &= (\omega p_i, \omega' q_i, r_i t_i, -(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i - r_i v_i, 0, \omega, \overline{[-d_i \omega, 0]})_{\mathbf{H}_i} \end{aligned}$$

$$\overline{\mathbf{e}(\overline{[\omega]}_2, \delta_i \cdot \overline{[\langle D, \mathbf{1} \rangle]_2})} = \overline{[\delta_i \langle D, \mathbf{1} \rangle \omega]}_{\mathbf{t}}$$

where  $\overline{[\langle D, \mathbf{1} \rangle]_2}$  comes from the public parameters.

Output  $\text{ct}_{\text{tag}, i} := (\mathbf{c}_{i, \text{ipfe}}, \overline{[\delta \cdot \mathbf{h}_{i,7}^*, \overline{[\delta_i \langle D, \mathbf{1} \rangle \omega]}_{\mathbf{t}}]})$ .

$\text{Dec}(\text{dk}_{\text{tag-f,y}}, \mathbf{c})$ : Parse

$$\begin{aligned} \text{dk}_{\text{tag-f,y}} &= ((\mathbf{k}_{i,\text{ipfe}}, \boxed{\epsilon \cdot \mathbf{h}_{i,8}})_i, \boxed{\llbracket \epsilon \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f,y}} \rrbracket_{\mathbf{t}}}); \\ \mathbf{c} &= (\mathbf{c}_{i,\text{ipfe}}, \boxed{\delta \cdot \mathbf{h}_{i,7}^*, \llbracket \delta_i \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}}})_{i=1}^n \end{aligned}$$

Compute  $\boxed{\llbracket \delta \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}}} = \sum_{i=1}^n \llbracket \delta_i \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}}$  and

$$\llbracket \text{out} \rrbracket_{\mathbf{t}} = \sum_{i=1}^n \left( (\text{ct}_{\text{tag},i} + \boxed{\epsilon \cdot \mathbf{h}_{i,8}}) \times (\mathbf{k}_{i,\text{ipfe}} + \boxed{\delta \cdot \mathbf{h}_{i,7}^*}) \right) + \boxed{\llbracket \epsilon \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f,y}} \rrbracket_{\mathbf{t}}} + \boxed{\llbracket \delta \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}}}.$$

Finally, compute the discrete logarithm and output the small value  $\text{out}$ .

The *correctness* of the scheme is verified by:

$$\begin{aligned} & \llbracket \text{out} \rrbracket_{\mathbf{t}} \\ &= \sum_{i=1}^n \left( (\mathbf{k}_{i,\text{ipfe}} + \delta \cdot \mathbf{h}_{i,7}^*) \times (\text{ct}_{\text{tag},i} + \epsilon \cdot \mathbf{h}_{i,8}) \right) + \boxed{\llbracket \epsilon \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f,y}} \rrbracket_{\mathbf{t}}} + \boxed{\llbracket \delta \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}}} \\ &= \sum_{i=1}^n \left( \begin{array}{c} (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, \frac{v_i}{t_i} y_i, y_i, \\ 0, \kappa_{\text{tag-f,y}} \theta_i, \delta, -e_i \kappa_{\text{tag-f,y}}) \mathbf{H}_i^* \\ \times \\ (\omega p_i, \omega' q_i, r_i t_i, -(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i - r_i v_i, \\ 0, \omega, -d_i \omega, \epsilon) \mathbf{H}_i \end{array} \right) \\ & \quad + \boxed{\llbracket \epsilon \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f,y}} \rrbracket_{\mathbf{t}}} + \boxed{\llbracket \delta \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}}} \\ & \stackrel{(*)}{=} \sum_{i=1}^n \boxed{\llbracket \omega \zeta_1 s_i y_i + \omega \zeta_4 u_i y_i + \omega' \zeta_2 s_i y_i + \omega' \zeta_3 u_i y_i + \theta_i \omega \kappa_{\text{tag-f,y}} \rrbracket_{\mathbf{t}}} \\ & \quad + \sum_{i=1}^n \boxed{\llbracket (-\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i \cdot y_i \rrbracket_{\mathbf{t}}} \\ & \quad + \sum_{i=1}^n \boxed{\llbracket (-\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i \cdot y_i \rrbracket_{\mathbf{t}}} \\ &= \boxed{\llbracket \langle \mathbf{x}, \mathbf{y} \rangle \rrbracket_{\mathbf{t}}} . \end{aligned}$$

where the equality (\*) comes from system (5). We recall that  $(\theta_i)_{i \in [n]}$  is a secret sharing of 0.

**Security.** We prove the one-time static security of our DMCFE scheme in the ROM, where the full-domain hash functions are modeled as random oracles, the sets of corrupted clients  $\mathcal{C}_{\text{ekey}}$  as well as  $\mathcal{C}_{\text{skey}}$  must be sent up front (*static* corruption), while the challenges  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  can be adaptively chosen (*adaptive* challenge). We note that we can achieve a better level of security in our concrete instantiation compared to the generic transformation. On one hand, our transformation follows the same blueprint in the work by Chotard *et al.* [CDSG<sup>+</sup>20], which is the most relevant to our DMCFE setting. We apply a layer of *All-or-Nothing Encapsulation* (AoNE) to our ciphertext and key components, which ensures that the original key/ciphertext components can be recovered only when all parts are gathered. Our concrete DMCFE in Section 6 builds the AoNE directly by combining one-time pad (OTP) and a random oracle (RO). Then, the programmability of the RO helps us circumvent the problem of adaptive queries. While programming the RO, we indeed exploits in a non-blackbox manner the OTP as a summation in  $\mathbb{Z}_q^*$  to accumulate a secret sharing of 0 on the honest parts (known in advance thanks to static corruption).

**Theorem 19.** *Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$  be the DMCFE constructed in Section 6.1. Then,  $\mathcal{E}$  is one-time statically IND+-secure in the ROM following the security model in Definition 18 if the SXDH and DBDH assumptions hold for  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . More specifically, let  $n$  denote the dimension for inner-products,  $Q_1, Q_2$  denote the maximum number of random*

oracle ( $RO$ ) queries to  $H_1, H_2$  and  $K$  denote the total number of functional key queries. For any one-time challenge ppt adversary  $\mathcal{A}$  against  $\mathcal{E}$  with static corruption of secret keys and encryption keys, we have the following bound:

$$\text{Adv}_{\mathcal{E}, \mathcal{F}^{\text{IP}}, \mathcal{A}}^{\text{dmc-stat-1chal}^+}(1^\lambda) \leq (K + 1)\text{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\text{DBDH}}(1^\lambda) + (3 + 2Q_1 + K)\text{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\text{SXDH}}(1^\lambda) + \frac{Q_2^2}{2q}.$$

Details are presented in the appendix [C.3](#).

## Acknowledgments

This work was supported in part by the France 2030 ANR Project ANR-22-PECY-003 Secure-Compute, the French ANR Project ANR-19-CE39-0011 PRESTO and the Beyond5G Project as part of the plan “France Relance”.

## References

- ABG19. Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT 2019, Part III*, 2019.
- ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In *PKC 2019, Part II*, 2019.
- ABN10. Michel Abdalla, Mihir Bellare, and Gregory Neven. Robust encryption. In *TCC 2010*, 2010.
- ABP<sup>+</sup>17. Shweta Agrawal, Sanjay Bhattacharjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In *ACM CCS 2017*, 2017.
- AGRW17. Michel Abdalla, Romain Gay, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT 2017, Part I*, 2017.
- AGT21a. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In *CRYPTO 2021, Part IV*, 2021.
- AGT21b. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In *TCC 2021, Part II*, 2021.
- AGT22. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *TCC '22*. Springer-Verlag, 2022. <https://ia.cr/2022/1168>.
- ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO 2016, Part III*, 2016.
- BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, 2001.
- BGH07. Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *48th FOCS*, 2007.
- BO13. Mihir Bellare and Adam O’Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS 13*, 2013.
- BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, 2011.
- CDG<sup>+</sup>18a. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT 2018, Part II*, 2018.
- CDG<sup>+</sup>18b. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. <https://eprint.iacr.org/2018/1021>.
- CDSG<sup>+</sup>20. Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In *CRYPTO 2020, Part I*, 2020.
- CLL<sup>+</sup>13. Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In *PAIRING 2012*, 2013.
- Coc01. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, 2001.
- DOT18. Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the  $k$ -Linear assumption. In *PKC 2018, Part II*, 2018.
- EHK<sup>+</sup>13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO 2013, Part II*, 2013.

- GGG<sup>+</sup>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 *EUROCRYPT 2014*, 2014.
- GKL<sup>+</sup>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>.
- LT19. Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT 2019, Part III*, 2019.
- LW10. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, 2010.
- OT10. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO 2010*, 2010.
- OT12a. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT 2012*, 2012.
- OT12b. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT 2012*, 2012.
- Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, 1984.
- SW05. Amit Sahai and Brent R. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, 2005.
- Wat09. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO 2009*, 2009.

## A Supporting Materials - Additional Definitions

### A.1 Decisional Separation Diffie-Hellman (DSDH) Assumption

**Definition 20.** *In a cyclic group  $\mathbb{G}$  of prime order  $q$ , the Decisional Separation Diffie-Hellman (DSDH) problem is to distinguish the distributions*

$$D_0 = \{(x, y, \llbracket 1 \rrbracket, \llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket ab + x \rrbracket\}\} \quad D_1 = \{(x, y, (\llbracket 1 \rrbracket, \llbracket a \rrbracket, \llbracket b \rrbracket, \llbracket ab + y \rrbracket))\}$$

for any  $x, y \in \mathbb{Z}_q$ , and  $a, b \xleftarrow{\$} \mathbb{Z}_q$ . The DSDH assumption in  $\mathbb{G}$  assumes that no ppt adversary can solve the DSDH problem with non-negligible probability.

### A.2 Dual Pairing Vector Spaces

**Basis changes.** In this work, we use extensively *basis changes* over dual orthogonal bases of a DPVS. We again use  $\mathbb{G}_1^N$  as a running example. Let  $(\mathbf{A}, \mathbf{A}^*)$  be the dual canonical bases of  $(\mathbb{G}_1^N, \mathbb{G}_2^N)$ . Let  $(\mathbf{U} = (\mathbf{u}_i)_i, \mathbf{U}^* = (\mathbf{u}_i^*)_i)$  be a pair of dual bases of  $(\mathbb{G}_1^N, \mathbb{G}_2^N)$ , corresponding to an invertible matrix  $U \in \mathbb{Z}_q^{N \times N}$ . Given an invertible matrix  $B \in \mathbb{Z}_q^{N \times N}$ , the basis change from  $\mathbf{U}$  w.r.t  $B$  is defined to be  $\mathbf{B} := B \cdot \mathbf{U}$ , which means:

$$\begin{aligned} (x_1, \dots, x_N)_{\mathbf{B}} &= \sum_{i=1}^N x_i \mathbf{b}_i = (x_1, \dots, x_N) \cdot \mathbf{B} = (x_1, \dots, x_N) \cdot B \cdot \mathbf{U} \\ &= (y_1, \dots, y_N)_{\mathbf{U}} \text{ where } (y_1, \dots, y_N) := (x_1, \dots, x_N) \cdot B \end{aligned}$$

Under a basis change  $\mathbf{B} = B \cdot \mathbf{U}$ , we have

$$(x_1, \dots, x_N)_{\mathbf{B}} = ((x_1, \dots, x_N) \cdot B)_{\mathbf{U}}; (y_1, \dots, y_N)_{\mathbf{U}} = \left( (y_1, \dots, y_N) \cdot B^{-1} \right)_{\mathbf{B}}.$$

The computation is extended to the dual basis change  $\mathbf{B}^* = B' \cdot \mathbf{U}^*$ , where  $B' = (B^{-1})^{\top}$ :

$$(x_1, \dots, x_N)_{\mathbf{B}^*} = ((x_1, \dots, x_N) \cdot B')_{\mathbf{U}^*}; (y_1, \dots, y_N)_{\mathbf{U}^*} = \left( (y_1, \dots, y_N) \cdot B^{\top} \right)_{\mathbf{B}^*}.$$

It can be checked that  $(\mathbf{B}, \mathbf{B}^*)$  remains a pair of dual orthogonal bases. When we consider a basis change  $\mathbf{B} = B \cdot \mathbf{U}$ , if  $B = (b_{i,j})_{i,j}$  affects only a subset  $J \subseteq [N]$  of indices in the representation w.r.t basis  $\mathbf{U}$ , we will write  $B$  as the square block containing  $(b_{i,j})_{i,j}$  for  $i, j \in J$  and implicitly the entries of  $B$  outside this block is taken from  $I_N$ .

## B Supporting Materials - Constructions

### B.1 Admissible Executions against DMCFE for Concrete Classes: Linear and Quadratic Functions

The goal of this section is to demonstrate that the optimality of our admissibility in Definition 6 holds for both important concrete function classes: the inner-product class and the class of quadratic functions. We will show that these two classes satisfy Definition 9, and at the same time translating the admissible conditions of Definition 6 in these particular cases.

**Admissibility for Inner-Product Function Class.** In the context of this paper, we will consider concrete constructions of DMCFE schemes for the class  $\mathcal{F}^{\text{IP}} = \{F_{\mathbf{y}}\}$  and  $F_{\mathbf{y}} : (\mathbb{Z}_q^*)^n \rightarrow \mathbb{Z}_q$ , to compute inner products over  $\mathbb{Z}_q$ . Their security will be proven in the security model of Definition 6 and in this section we translate the admissibility of adversaries against DMCFE schemes for the class  $\mathcal{F}^{\text{IP}}$ . As discussed in Remark 4, the parameters of  $F_{\mathbf{y}} \in \mathcal{F}^{\text{IP}}$  are simply the vector  $\mathbf{y} \in \mathbb{Z}_q^n$ . Hence the number of secret keys and of encryption keys are equal to  $n$ . We prove the following claim on the efficient decidability and fixed-component indistinguishability of  $\mathcal{F}^{\text{IP}}$ , using the linearity of inner products over  $\mathbb{Z}_q$ . These admissibility conditions will be crucially used in the proof of security for our constructions in Section 5.1 and Section 6.

**Theorem 21.** *Let  $\lambda \in \mathbb{N}$  and we define  $\mathcal{F}^{\text{IP}} = \{F_{\mathbf{y}}\}$  where  $F_{\mathbf{y}} : (\mathbb{Z}_q^*)^n \rightarrow \mathbb{Z}_q$  w.r.t  $\lambda$ . Then, the class  $\mathcal{F}^{\text{IP}}$  is fixed-component distinguishable with efficiently decidable admissibility.*

*Proof.* Let  $\mathcal{H}_{\text{skey}}, \mathcal{H}_{\text{ekey}}$  be the sets of honest clients considered in **Finalize** in Figure 1 and let  $\mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}$  be their complementary sets of corrupted clients. We consider the challenge messages  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  and a functional key corresponding to  $\mathbf{y} \in \mathbb{Z}_q^n$ , under relation (4) in Definition 6. We have to interpret the constraints in the particular case of inner products. First of all, for all  $i \in \mathcal{C}_{\text{skey}} \cup \mathcal{C}_{\text{ekey}}$ , independent of the queries to **LoR** for the challenges  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  or **DKeyGenShare** for  $\mathbf{y}$ , an admissible adversary can use  $\text{ek}_i$  or  $\text{sk}_i$  to craft their own ciphertext or key components. We emphasize that the order of ciphertext or key queries, for which the adversary is supposed to get the whole  $n$  components, and the corruption the clients over  $\text{ek}_i$  or  $\text{sk}_i$ , can be arbitrary.

During **Finalize**, we take into account all *deducible* challenge messages and functions. Let  $\mathcal{C} := \mathcal{C}_{\text{skey}} \cup \mathcal{C}_{\text{ekey}}$  and  $\mathcal{H} := [n] \setminus \mathcal{C}$ . In particular, the functional key allows decrypting the challenge ciphertext and it must be the case that  $\langle \Delta \mathbf{x}, \mathbf{y} \rangle = 0$ , where  $\Delta \mathbf{x} := \mathbf{x}_0^* - \mathbf{x}_1^*$  for the initial  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  and  $\mathbf{y}$  is an inner-product function queried to **DKeyGenShare**. For  $b \in \{0, 1\}$ , we define  $\mathbf{x}_{\text{ekey}}^{(b)} := (\mathbf{x}_b^*[i] : i \in \mathcal{H}_{\text{ekey}})$  and let  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  be an arbitrary pair of vectors deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$ . On the other hand, let  $G$  be a function with parameters  $\mathbf{y}_G$  deducible from  $\mathbf{y}_{\text{skey}} := (\mathbf{y}[i] : i \in \mathcal{H}_{\text{skey}})$ . We consider admissible adversaries, *i.e.*  $\langle \Delta \mathbf{z}, \mathbf{y}_G \rangle = 0$  for all  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  and all  $\mathbf{y}_G$ . Expanding the formula of  $\langle \Delta \mathbf{z}, \mathbf{y}_G \rangle$  by  $i$  gives:

$$\begin{aligned} 0 &= \langle \Delta \mathbf{z}, \mathbf{y}_G \rangle \\ &= \sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] + \sum_{i \in (\mathcal{C}_{\text{skey}} \setminus \mathcal{C}_{\text{ekey}})} \Delta \mathbf{x}[i] \mathbf{y}_G[i] \\ &\quad + \sum_{i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}} (\mathbf{z}^{(0)}[i] - \mathbf{z}^{(1)}[i]) \cdot \mathbf{y}[i] + \sum_{i \in \mathcal{C}_{\text{ekey}} \cap \mathcal{C}_{\text{skey}}} (\mathbf{z}^{(0)}[i] - \mathbf{z}^{(1)}[i]) \cdot \mathbf{y}_G[i] . \end{aligned}$$

We notice that this must hold for *every* deducible function  $G$  that computes the inner products with  $\mathbf{y}_G$  and *every* deducible  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$ , which makes the above imply

- For every  $i \in \mathcal{C}_{\text{skey}} \setminus \mathcal{C}_{\text{ekey}}$ , it must hold that  $\Delta \mathbf{x}[i] = 0$  because  $\mathbf{y}_G[i]$  is fully controlled by the adversary even after receiving the functional key. Otherwise the adversary can query  $(\mathbf{x}_0^*[i], \mathbf{x}_1^*[i], \text{tag})$  to **LoR** where  $\mathbf{x}_0^*[i] \neq \mathbf{x}_1^*[i]$ , and deduce  $\mathbf{y}_G$  so that the linear combination  $\sum_{i \in (\mathcal{C}_{\text{skey}} \setminus \mathcal{C}_{\text{ekey}})} \Delta \mathbf{x}[i] \mathbf{y}_G[i]$  is not 0, while setting the rest to 0.



- For every  $i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ , it must hold that  $(\mathbf{z}^{(0)}[i] - \mathbf{z}^{(1)}[i]) \cdot \mathbf{y}[i] = 0$  because  $\mathbf{z}^{(0)}[i] - \mathbf{z}^{(1)}[i]$  is fully controlled by the adversary even after receiving the challenge. Otherwise, the adversary can make queries  $(\mathbf{x}_0^*[i], \mathbf{x}_1^*[i], \text{tag})$  to  $\mathbf{LoR}$  for  $i \in \mathcal{H}_{\text{ekey}}$ , then deduce  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$ , so that  $\sum_{i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}} (\mathbf{z}^{(0)}[i] - \mathbf{z}^{(1)}[i]) \cdot \mathbf{y}[i] \neq 0$  for some functional key query of  $\mathbf{y}$  while setting the rest to 0, which violates the 0-equality. Since  $(\mathbf{z}^{(0)}[i], \mathbf{z}^{(1)}[i])$  are deducible coordinates, they can be the original challenge coordinates  $(\mathbf{x}_0^*[i], \mathbf{x}_1^*[i])$ .
- For every  $i \in \mathcal{C}_{\text{ekey}} \cap \mathcal{C}_{\text{skey}}$ , it must hold that  $(\mathbf{z}^{(0)}[i] - \mathbf{z}^{(1)}[i]) \cdot \mathbf{y}_G[i] = 0$  because both the input's term as well as the function's term are fully controlled by the adversary. This gives  $\mathbf{x}_0^*[i] = \mathbf{x}_1^*[i]$  in particular, when we consider the original challenge coordinates and the function having  $\mathbf{y}_G[i] = 1$ .
- Finally, we must also have  $\sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] = 0$ .

In the end, in the case of DMCFE for inner products, the condition in Definition 6 for any admissible attack in the security game is expressed as:

1. For all  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$ ,  $\sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] = 0$  where  $\Delta \mathbf{x}[i] = \mathbf{x}_0^*[i^*] - \mathbf{x}_1^*[i^*]$ .
2. For all  $i^* \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ , either  $\mathbf{x}_0^*[i^*] = \mathbf{x}_1^*[i^*] = 0$  or  $\mathbf{y}[i^*] = 0$ .
3. For all  $i^* \in \mathcal{C}_{\text{skey}}$ ,  $\mathbf{x}_0^*[i^*] = \mathbf{x}_1^*[i^*]$ .

It is clear that these conditions can be efficiently checked during **Finalize**.

Next, it can be verified that  $\mathcal{F}^{\text{IP}}$  is *fixed-component distinguishable*. We consider two cases: the zero function  $F_0$  and non-constant functions. First of all, the parameter vector of  $F_0$  is  $\mathbf{0} = (0, \dots, 0)$  and by setting  $\mathcal{H}_{\text{func}} = \{2, 3, \dots, n\}$ , the function  $G$  having parameters  $\mathbf{y}_G = (1, 0, \dots, 0)$  is deducible from  $\mathbf{0}_{\text{func}} = (0)_{i \in \mathcal{H}_{\text{func}}}$  and is fixed-component distinguishable. An example of a triple that distinguishes  $G$  is  $\mathcal{H}_{\text{inp}} = \{1\}$  and  $\mathbf{x}_{\text{inp}}^{(0)}[1] \neq \mathbf{x}_{\text{inp}}^{(1)}[1]$  are some fixed values, with a deducible pair  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  where  $\mathbf{x}_{\text{inp}}^{(0)}[1] = \mathbf{z}^{(0)}[1] \neq \mathbf{z}^{(1)}[1] = \mathbf{x}_{\text{inp}}^{(1)}[1]$ ,  $\mathbf{z}^{(b)}[i] = \mathbf{z}^{(1-b)}[i] = 0$  for all  $i \geq 2, b \in \{0, 1\}$ .

Let  $F_{\mathbf{y}}(\cdot) = \langle \cdot, \mathbf{y} \rangle$  be a non-constant function in  $\mathcal{F}^{\text{IP}}$ , then there exists  $I \subseteq [n]$  such that  $\mathbf{y}[i] \neq 0$  for  $i \in I$ . A triple that distinguishes  $F_{\mathbf{y}}$  contains  $\mathcal{H}_{\text{inp}} = I$ . We then sample uniformly at random, for  $i \in I$ ,  $\mathbf{x}_{\text{inp}}^{(1)}[i] \xleftarrow{\$} \mathbb{Z}_q$ . Afterwards, we choose  $i^* \in I$  and for  $i \in I \setminus \{i^*\}$ , sample  $\Delta \mathbf{x}[i] \xleftarrow{\$} \mathbb{Z}_q$  and define  $\mathbf{x}_{\text{inp}}^{(0)}[i] := \mathbf{x}_{\text{inp}}^{(1)}[i] - \Delta \mathbf{x}[i]$ . Then, we sample uniformly at random a nonzero  $z \xleftarrow{\$} \mathbb{Z}_q^*$  and solve for  $\Delta \mathbf{x}[i^*]$  from the equation

$$z - \sum_{i \in I \setminus \{i^*\}} \Delta \mathbf{x}[i] \mathbf{y}[i] = \Delta \mathbf{x}[i^*] \mathbf{y}[i^*] .$$

Since  $i^* \in I$ ,  $\mathbf{y}[i^*] \neq 0$  and there is a unique solution for  $\Delta \mathbf{x}[i^*]$  that can be efficiently found. The witness is  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  where  $\mathbf{z}^{(b)}[i] = \mathbf{x}_{\text{inp}}^{(b)}[i]$  for all  $i \in \mathcal{H}_{\text{inp}} = I$ ,  $\mathbf{z}^{(b)}[i] = \mathbf{z}^{(b)}[i] = 0$  for all  $i \in [n] \setminus I, b \in \{0, 1\}$ . This concludes the fixed-component distinguishability of  $\mathcal{F}^{\text{IP}}$  over  $\mathbb{Z}_q$ .  $\square$

**Remark 22.** As we see in Remark 15, the admissibility *as per* Definition 6 is optimal for  $\mathcal{F}^{\text{IP}}$  over  $\mathbb{Z}_q$  and Theorem 23 shows the same holds for  $\mathcal{F}_{B, B'}^{\text{IP}, \text{poly}}$ . Sections 5 and 6 give constructions of DMCFE for  $\mathcal{F}_{B, B'}^{\text{IP}, \text{poly}}$  that are finally secure in the security model under the translated admissibility, demonstrating the feasibility of our security notion from Definition 7.

**Quadratic Functions is Fixed-component distinguishable, with Efficiently Decidable Admissibility.** We show that the class  $\mathcal{F}^{\text{quad}}$  of quadratic functions  $f_{\mathbf{c}}(\mathbf{x}) := \langle \mathbf{x} \otimes \mathbf{x}, \mathbf{c} \rangle$  for  $\mathbf{x} \in \mathbb{Z}_q^n$  as studied in [AGT21a], where the function's parameters are  $\mathbf{c} = (c_{i,j})_{i,j \in [n]} \in \mathbb{Z}_q^{n^2}$ , is *fixed-component distinguishable*, following Definition 9, thanks to the fact that we can solve quadratic equations modulo a prime efficiently. The case of the constant function, which maps every input to 0, can be handled similarly as in the case of inner-products. Given a non-constant  $f_{\mathbf{c}} \in \mathcal{F}^{\text{quad}}$ , in particular  $f_{\mathbf{c}}$  is non-zero, there must exist  $i \in [n]$  such that the coefficients

$(c_{i,j})_{j \neq i}, (c_{\ell,i})_{\ell \neq i}, c_{i,i}$  are not all zeros. We first sample  $\mathbf{x}[j] \xleftarrow{\$} \mathbb{Z}_q^*$  uniformly at random, for all  $j \neq i$ . Then, we solve the following quadratic equation for  $\mathbf{x}'[i]$ , for some  $z \neq 0$ :

$$z = (\mathbf{x}'[i] - \mathbf{x}[i]) \cdot \left( \sum_{j \neq i} \mathbf{x}[j] c_{i,j} + \sum_{\ell \neq i} \mathbf{x}[\ell] c_{\ell,i} \right) + c_{i,i} \cdot (\mathbf{x}'[i]^2 - \mathbf{x}[i]^2). \quad (6)$$

Thanks to the choice of  $i$ , *i.e.* the coefficients  $(c_{i,j})_{j \neq i}, (c_{\ell,i})_{\ell \neq i}, c_{i,i}$  are not all zeros, and the fact that  $\mathbf{x}[j] \xleftarrow{\$} \mathbb{Z}_q^*$  for all  $j \neq i$ , with probability  $1/2$  the equation (6) will have a solution  $\mathbf{x}'[i]$  that can be found efficiently using Tonelli-Shanks algorithm. In the end, setting  $\mathbf{x}' := (\mathbf{x}[1], \dots, \mathbf{x}[i-1], \mathbf{x}'[i], \mathbf{x}[i+1], \dots, \mathbf{x}[n])$  verifies  $f_{\mathbf{c}}(\mathbf{x}) \neq f_{\mathbf{c}}(\mathbf{x}')$ . Hence, a triple that distinguishes  $f_{\mathbf{c}}$  is  $(\mathbf{x}, \mathbf{x}', \mathcal{H}_{\text{ekey}} := [n])$  and its witness is  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)}) = (\mathbf{x}, \mathbf{x}')$ .

We now move on to the translation of the admissibility condition in Definition 6 in the case of  $\mathcal{F}^{\text{quad}}$ . During **Finalize**, we take into account all *deducible* challenge messages and functions. Let  $\mathcal{C}_{\text{ekey}} \subset [n]$  and  $\mathcal{H}_{\text{ekey}} := [n] \setminus \mathcal{C}_{\text{ekey}}$ . Let  $\mathcal{C}_{\text{skey}} \subset [n] \times [n]$  and  $\mathcal{H}_{\text{skey}} := [n] \times [n] \setminus \mathcal{C}_{\text{skey}}$ . In particular, the functional key allows decrypting the challenge ciphertext and it must be the case that  $\langle \Delta \mathbf{x}, \mathbf{c} \rangle = 0$ , where  $\Delta \mathbf{x} := \mathbf{x}_0^* \otimes \mathbf{x}_0^* - \mathbf{x}_1^* \otimes \mathbf{x}_1^*$  for the initial  $(\mathbf{x}_0^*, \mathbf{x}_1^*)$  and  $\mathbf{c}$  is the parameter vector of a quadratic function  $f_{\mathbf{c}}$  queried to **DKeyGenShare**. For  $b \in \{0, 1\}$ , we define  $\mathbf{x}_{\text{ekey}}^{(b)} := (\mathbf{x}_b^*[i] : i \in \mathcal{H}_{\text{ekey}})$  and let  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  be an arbitrary pair of vectors deducible from  $(\mathbf{x}_{\text{ekey}}^{(0)}, \mathbf{x}_{\text{ekey}}^{(1)})$ . On the other hand, let  $G$  be a function deducible from  $\mathbf{c}_{\text{skey}} := (c_{i,j} : (i,j) \in \mathcal{H}_{\text{skey}})$ , having parameters  $\mathbf{c}^G = (c_{i,j}^G)_{i,j} \in \mathbb{Z}_q^{n^2}$ . We consider admissible attacks in the security game, *i.e.*  $\langle \Delta \mathbf{z}, \mathbf{c}^G \rangle = 0$  for all  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$  and all  $\mathbf{c}^G$ . Expanding the formula of  $\langle \Delta \mathbf{z}, \mathbf{c}^G \rangle$  gives:

$$\begin{aligned} 0 &= \langle \Delta \mathbf{z}, \mathbf{c}^G \rangle \\ &= \sum_{i,j \in \mathcal{C}_{\text{ekey}}} (\mathbf{z}_0^*[i] \mathbf{z}_0^*[j] - \mathbf{z}_1^*[i] \mathbf{z}_1^*[j]) c_{i,j}^G + \sum_{\substack{i,j \in \mathcal{H}_{\text{ekey}} \\ (i,j) \notin \mathcal{C}_{\text{skey}}}} (\mathbf{x}_0^*[i] \mathbf{x}_0^*[j] - \mathbf{x}_1^*[i] \mathbf{x}_1^*[j]) c_{i,j} \\ &\quad + \sum_{\substack{i,j \in \mathcal{H}_{\text{ekey}} \\ (i,j) \in \mathcal{C}_{\text{skey}}}} (\mathbf{x}_0^*[i] \mathbf{x}_0^*[j] - \mathbf{x}_1^*[i] \mathbf{x}_1^*[j]) c_{i,j}^G + \sum_{i \in \mathcal{C}_{\text{ekey}}} \sum_{j \in \mathcal{H}_{\text{ekey}}} (\mathbf{z}^{(0)}[i] \mathbf{x}_0^*[j] - \mathbf{z}^{(1)}[i] \mathbf{x}_1^*[j]) c_{i,j}^G \\ &\quad + \sum_{i \in \mathcal{H}_{\text{ekey}}} \sum_{j \in \mathcal{C}_{\text{ekey}}} (\mathbf{x}_0^*[i] \mathbf{z}^{(0)}[j] - \mathbf{x}_1^*[i] \mathbf{z}^{(1)}[j]) c_{i,j}^G. \end{aligned}$$

We are using the fact that if  $(i,j) \notin \mathcal{C}_{\text{skey}}$  then  $c_{i,j}^G = c_{i,j}$ . We notice that this must hold for *every* deducible function  $G$  that computes the quadratic function with  $\mathbf{c}^G$  and *every* deducible  $(\mathbf{z}^{(0)}, \mathbf{z}^{(1)})$ , which makes the above imply

- First of all, for all  $i, j \in \mathcal{H}_{\text{ekey}}$  such that  $(i,j) \in \mathcal{C}_{\text{skey}}$ , because  $c_{i,j}^G$  can be fully changed by the adversary even after receiving the functional key, it must hold that  $\mathbf{x}_0^*[i] \mathbf{x}_0^*[j] = \mathbf{x}_1^*[i] \mathbf{x}_1^*[j] \forall i, j \in \mathcal{H}_{\text{ekey}}, (i,j) \in \mathcal{C}_{\text{skey}}$ .
- For each  $j \in \mathcal{H}_{\text{ekey}}$ , if there exists  $i \in \mathcal{C}_{\text{ekey}}$  such that  $(i,j) \in \mathcal{C}_{\text{skey}}$  (resp.  $(j,i) \in \mathcal{C}_{\text{skey}}$ ), then it must hold that  $(\mathbf{z}^{(0)}[i] \mathbf{x}_0^*[j] - \mathbf{z}^{(1)}[i] \mathbf{x}_1^*[j]) c_{i,j}^G = 0$  since the adversary can put whatever he wants into  $(\mathbf{z}^{(0)}[i], \mathbf{z}^{(1)}[i], c_{i,j}^G)$  even after the querying phase. This particularly makes  $\mathbf{x}_0^*[i] \mathbf{x}_0^*[j] = \mathbf{x}_1^*[i] \mathbf{x}_1^*[j]$  as we can set  $\mathbf{z}^{(0)}[i], \mathbf{z}^{(1)}[i]$  to the original challenge coordinates and  $c_{i,j}^G = 1$ .
- For every  $i, j \in \mathcal{C}_{\text{ekey}}$  such that  $(i,j) \in \mathcal{C}_{\text{skey}}$ , the admissibility dictates that we need to restrain on individual pairs  $(i,j)$ , because the adversary can manipulate either the challenge part or the key parts or both otherwise. The constraint we must employ is for every such pair  $(i,j)$ , we have  $(\mathbf{z}_0^*[i] \mathbf{z}_0^*[j] - \mathbf{z}_1^*[i] \mathbf{z}_1^*[j]) c_{i,j}^G = 0$ . This particularly makes  $\mathbf{x}_0^*[i] \mathbf{x}_0^*[j] = \mathbf{x}_1^*[i] \mathbf{x}_1^*[j]$  as we can set  $c_{i,j}^G = 1$  and  $\mathbf{z}^{(0)}[i], \mathbf{z}^{(1)}[i]$  to the original challenge coordinates.

– Finally, we must also impose

$$\sum_{\substack{i,j \in \mathcal{H}_{\text{ekey}} \\ (i,j) \notin \mathcal{C}_{\text{skey}}}} (\mathbf{x}_0^*[i]\mathbf{x}_0^*[j] - \mathbf{x}_1^*[i]\mathbf{x}_1^*[j])c_{i,j} = 0 .$$

These conditions can be checked efficiently, hence the class  $\mathcal{F}^{\text{quad}}$  has efficiently decidable admissibility. Moreover, we have also translated the admissibility condition as defined in Definition 6 for the class  $\mathcal{F}^{\text{quad}}$ , implying that it has efficient decidability. As a result, the optimality of our admissibility also holds for  $\mathcal{F}^{\text{quad}}$  over  $\mathbb{Z}_q$  following Theorem 12.

**Optimality in the Case of Inner Products in Polynomially Bounded Range.** In our concrete DDH-based construction of DMCFE for inner products in Section 5.1 and Section 6, our functionality is not for inner products over  $\mathbb{Z}_q^n$  set up w.r.t  $\lambda \in \mathbb{N}$ , but only for *bounded* vectors such that the inner product evaluation is polynomially large. More specifically, we name this class  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$  in which any function  $f_{\mathbf{y}} : \mathbf{x} \mapsto \langle \mathbf{x}, \mathbf{y} \rangle$  receives as inputs  $\mathbf{x}$  and has parameters  $\mathbf{y}$  such that  $\|\mathbf{x}\|_{\infty} < B$  and  $\|\mathbf{y}\|_{\infty} < B'$ , where  $B, B' = \text{poly}(\lambda) \in \mathbb{N}$  are polynomials. The concrete admissibility (see Remark 15) is still the same because we are still computing inner products. Below we prove that the admissibility *as per* Definition 6 is optimal for  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$ .

**Theorem 23.** *Let  $\lambda \in \mathbb{N}$  and  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$  be the function class to compute inner products in ranges parametrized by  $B, B'$ . Then, our admissibility condition as defined in Definition 6 is optimal for  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$ .*

*Proof.* Without loss of generality, we consider the one-challenge notion. We need to prove that: there exists a ppt distinguisher  $\mathcal{S}$  so that for any non-admissible attack of an adversary  $\mathcal{A}$  against some DMCFE  $\mathcal{E}$  for  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$  in a security experiment  $\text{Expr}_{\mathcal{E}, \mathcal{F}_{B,B'}^{\text{IP,poly}}, \mathcal{A}}$  given in Figure 1, we have

$$\Pr [\mathcal{S}(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\}) = b : b \leftarrow \text{Chall}(\text{rand}_{\text{Chall}})] \geq \frac{1}{\text{poly}(\lambda)} .$$

Any non-admissible attack will make one of the following hold:

1. There exists  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$  s.t  $\sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] \neq 0$ .
2. There exists  $i^* \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$  s.t  $(\mathbf{x}^{(0)}[i^*] - \mathbf{x}^{(1)}[i^*]) \mathbf{y}[i^*] \neq 0$ .
3. There exists  $i^* \in \mathcal{C}_{\text{skey}}$  s.t  $\mathbf{x}^{(0)}[i^*] \neq \mathbf{x}^{(1)}[i^*]$ .

We describe the distinguisher  $\mathcal{S}$  as follows and specify the strategy for each case:

1. The distinguisher  $\mathcal{S}$  parses

$$(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}, \{(\mathbf{x}, \text{tag}^{(k)})\})$$

then use  $\mathcal{E}^{\text{abs}}$  and  $(\mathcal{Q}, \mathcal{Q}_{\text{Enc}}, \{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\})$  for abstracting the key components to obtain  $\{(\text{dk}_{\text{tag-f}, F, j}^{\text{abs}})_{j \in [m]}\}_{(\text{tag-f}, F) \in \mathcal{Q}}$ , the challenge ciphertext components to obtain  $(\text{ct}_{\text{tag}, i}^{\text{abs}})_{i \in [n]}$  for each  $\{(\mathbf{x}_0^*, \mathbf{x}_1^*, \text{tag})\}$ , and the encryption responses to obtain  $(\text{ct}_i^{\text{abs}, (k)})_{i \in [n]}$ . If there are corrupted keys  $\text{sk}_j$  or  $\text{ek}_i$  queried by  $\mathcal{A}$ , they will also be replaced by their abstracted counterparts  $\text{sk}_j^{\text{abs}}$  or  $\text{ek}_i^{\text{abs}}$ . In the following  $\mathcal{S}$  only needs the abstract DMCFE for  $\mathcal{F}_{B,B'}^{\text{IP,poly}}$

$$\mathcal{E}^{\text{abs}} = (\text{Setup}^{\text{abs}}, \text{DKShare}^{\text{abs}}, \text{DKeyComb}^{\text{abs}}, \text{Enc}^{\text{abs}}, \text{Dec}^{\text{abs}})$$

that satisfies the correctness requirement, no matter what the details of the concrete  $\mathcal{E}$  are.

2. If there exists  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$  such that  $\langle \mathbf{x}_0^*, \mathbf{y} \rangle \neq \langle \mathbf{x}_1^*, \mathbf{y} \rangle$ ,  $\mathcal{S}$  combines the key components of  $(\text{tag-f}, \mathbf{y})$ , decrypts the challenge ciphertext components, and outputs 1 if and only if the result is  $\langle \mathbf{x}_1^*, \mathbf{y} \rangle$ . All algorithms come from  $\mathcal{E}^{\text{abs}} = (\text{Setup}^{\text{abs}}, \text{DKShare}^{\text{abs}}, \text{DKeyComb}^{\text{abs}}, \text{Enc}^{\text{abs}}, \text{Dec}^{\text{abs}})$ . Else, in the following we assume that  $\langle \mathbf{x}_0^*, \mathbf{y} \rangle = \langle \mathbf{x}_1^*, \mathbf{y} \rangle$  for all  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$ .

3. If case 1 happens:
  - Let  $(\text{tag-f}, \mathbf{y}) \in \mathcal{Q}$  be the query such that  $\sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] \neq 0$ .
  - $\mathcal{S}$  uses the corrupted secret keys  $(\text{sk}_i)_{i \in \mathcal{C}_{\text{skey}}}$  and the honest component  $(\text{dk}_{\text{tag-f}, i})_{i \in \mathcal{H}_{\text{skey}}}$  to compute the partial functional keys  $(\tilde{\text{dk}}_{\text{tag-f}, i})_{i=1}^n$  for  $\tilde{\mathbf{y}}$  where  $(\tilde{\mathbf{y}}[i])_{i \in \mathcal{H}_{\text{skey}}} := (\mathbf{y}[i])_{i \in \mathcal{H}_{\text{skey}}}$  and  $(\tilde{\mathbf{y}}[i])_{i \in \mathcal{C}_{\text{skey}}} := \mathbf{0}$ . The key derivation algorithm is the abstract algorithm  $\text{DKShare}^{\text{abs}}$ .
  - $\mathcal{S}$  uses the corrupted encryption keys  $(\text{ek}_i)_{i \in \mathcal{C}_{\text{ekey}}}$  and the honest challenge ciphertext components  $(\text{ct}_{\text{tag}, i})_{i \in \mathcal{H}_{\text{ekey}}}$  to compute the ciphertext components  $(\tilde{\text{ct}}_{\text{tag}, i})_{i=1}^n$  for  $\tilde{\mathbf{x}}$  where implicitly  $(\tilde{\mathbf{x}}[i])_{i \in \mathcal{H}_{\text{ekey}}} := (\mathbf{x}_b^*[i])_{i \in \mathcal{H}_{\text{ekey}}}$  and  $(\tilde{\mathbf{x}}[i])_{i \in \mathcal{C}_{\text{ekey}}} := \mathbf{0}$ . The encryption is done using the abstract algorithm  $\text{Enc}^{\text{abs}}$ .
  - By using  $\text{DKeyComb}^{\text{abs}}$  to combine the newly generated key components  $(\tilde{\text{dk}}_{\text{tag-f}, i})_{i=1}^n$  and  $\text{Dec}^{\text{abs}}$  to decrypt the newly generated challenge ciphertext  $(\tilde{\text{ct}}_{\text{tag}, i})_{i \in [n]}$ ,  $\mathcal{S}$  outputs 1 if and only if the result is equal to  $\sum_{i \in \mathcal{H}} \mathbf{x}_1^*[i] \mathbf{y}[i]$ .
4. If case 2 happens:
  - If case 1 also happens,  $\mathcal{S}$  operates as above.
  - Else, let  $i^* \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$  s.t.  $(\mathbf{x}^{(0)}[i^*] - \mathbf{x}^{(1)}[i^*]) \mathbf{y}[i^*] \neq 0$ .
  - $\mathcal{S}$  uses the corrupted encryption keys  $(\text{ek}_i)_{i \in \mathcal{C}_{\text{ekey}} \setminus \{i^*\}}$  and the honest challenge ciphertext components  $(\text{ct}_{\text{tag}, i})_{i \in \mathcal{H}_{\text{ekey}}}$  as well as  $\text{ct}_{\text{tag}, i^*}$  to compute the ciphertext components  $(\tilde{\text{ct}}_{\text{tag}, i})_{i=1}^n$  for  $\tilde{\mathbf{x}}$  where implicitly  $(\tilde{\mathbf{x}}[i])_{i \in \mathcal{H}_{\text{ekey}}} := (\mathbf{x}_b^*[i])_{i \in \mathcal{H}_{\text{ekey}}}$ ,  $(\tilde{\mathbf{x}}[i])_{i \in \mathcal{C}_{\text{ekey}} \setminus \{i^*\}} := \mathbf{0}$ , and implicitly  $\tilde{\mathbf{x}}[i^*] := \mathbf{x}_b^*[i^*]$ . The encryption is done using the abstract algorithm  $\text{Enc}^{\text{abs}}$ .
  - $\mathcal{S}$  uses the corrupted secret keys  $(\text{sk}_i)_{i \in \mathcal{C}_{\text{skey}}}$  and the honest component  $(\text{dk}_{\text{tag-f}, i})_{i \in \mathcal{H}_{\text{skey}}}$  to compute the partial functional keys  $(\tilde{\text{dk}}_{\text{tag-f}, i})_{i=1}^n$  for  $\tilde{\mathbf{y}}$  where  $(\tilde{\mathbf{y}}[i])_{i \in \mathcal{H}_{\text{skey}}} := (\mathbf{y}[i])_{i \in \mathcal{H}_{\text{skey}}}$  and  $(\tilde{\mathbf{y}}[i])_{i \in \mathcal{C}_{\text{skey}}} := \mathbf{0}$ . We notice that implicitly  $\tilde{\mathbf{y}}[i^*] := \mathbf{y}[i^*]$  because  $i^* \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ . The key derivation algorithm is the abstract algorithm  $\text{DKShare}^{\text{abs}}$ .
  - By using  $\text{DKeyComb}^{\text{abs}}$  to combine the newly generated key components  $(\tilde{\text{dk}}_{\text{tag-f}, i})_{i=1}^n$  and  $\text{Dec}^{\text{abs}}$  to decrypt the newly generated challenge ciphertext  $(\tilde{\text{ct}}_{\text{tag}, i})_{i \in [n]}$ ,  $\mathcal{S}$  outputs 1 if and only if the result is equal to  $\mathbf{x}_1^*[i^*] \mathbf{y}[i^*]$ .
5. If case 3 happens:
  - If case 1 or case 2 also happens,  $\mathcal{S}$  operates as above.
  - Else, let  $i^* \in \mathcal{C}_{\text{skey}}$  s.t.  $\mathbf{x}_0^*[i^*] \neq \mathbf{x}_1^*[i^*]$ .
  - $\mathcal{S}$  uses the corrupted secret keys  $(\text{sk}_i)_{i \in \mathcal{C}_{\text{skey}} \setminus \{i^*\}}$  and the honest component  $(\text{dk}_{\text{tag-f}, i})_{i \in \mathcal{H}_{\text{skey}}}$  to compute the partial functional keys  $(\tilde{\text{dk}}_{\text{tag-f}, i})_{i=1}^n$  for  $\tilde{\mathbf{y}}$  where  $(\tilde{\mathbf{y}}[i])_{i \in \mathcal{H}_{\text{skey}}} := (\mathbf{y}[i])_{i \in \mathcal{H}_{\text{skey}}}$  and  $(\tilde{\mathbf{y}}[i])_{i \in \mathcal{C}_{\text{skey}} \setminus \{i^*\}} := \mathbf{0}$ . If  $\mathbf{y}[i^*] = 0$  the distinguisher  $\mathcal{S}$  sets  $\tilde{\mathbf{y}}[i^*] = z$  for some  $B' > z \neq 0$  and uses  $\text{sk}_{i^*}$  to compute  $\tilde{\text{dk}}_{\text{tag-f}, i^*}$ . The key derivation algorithm is the abstract algorithm  $\text{DKShare}^{\text{abs}}$ .
  - $\mathcal{S}$  uses the corrupted encryption keys  $(\text{ek}_i)_{i \in \mathcal{C}_{\text{ekey}} \setminus \{i^*\}}$  and the honest challenge ciphertext components  $(\text{ct}_{\text{tag}, i})_{i \in \mathcal{H}_{\text{ekey}}}$  as well as  $\text{ct}_{\text{tag}, i^*}$  to compute the ciphertext components  $(\tilde{\text{ct}}_{\text{tag}, i})_{i=1}^n$  for  $\tilde{\mathbf{x}}$  where implicitly  $(\tilde{\mathbf{x}}[i])_{i \in \mathcal{H}_{\text{ekey}}} := (\mathbf{x}_b^*[i])_{i \in \mathcal{H}_{\text{ekey}}}$ ,  $(\tilde{\mathbf{x}}[i])_{i \in \mathcal{C}_{\text{ekey}} \setminus \{i^*\}} := \mathbf{0}$ , and implicitly  $\tilde{\mathbf{x}}[i^*] := \mathbf{x}_b^*[i^*]$ . The encryption is done using the abstract algorithm  $\text{Enc}^{\text{abs}}$ .
  - By using  $\text{DKeyComb}^{\text{abs}}$  to combine the newly generated key components  $(\tilde{\text{dk}}_{\text{tag-f}, i})_{i=1}^n$  and  $\text{Dec}^{\text{abs}}$  to decrypt the newly generated challenge ciphertext  $(\tilde{\text{ct}}_{\text{tag}, i})_{i \in [n]}$ ,  $\mathcal{S}$  outputs 1 if and only if the result is equal to  $\mathbf{x}_1^*[i^*] \mathbf{y}[i^*]$ .

The condition in step 2 can be checked efficiently as  $\mathcal{A}$  is ppt and  $\mathcal{Q}$  must thus be of polynomial size. It can be verified that in all three cases the distinguisher  $\mathcal{S}$  outputs 1 if and only if the attack corresponds to an execution in which the challenge picks 1 as the challenge bit, for *any* DMCFE scheme  $\mathcal{E}'$ . The proof is completed.  $\square$

## B.2 Generic DMCFE for Inner-Products in the Stronger Model against Incomplete Queries

In previous works, one can transform an MIFE/MCFE that is secure against only complete queries to an MIFE/MCFE that is secure against also incomplete queries, *e.g.* see [AGRW17, CDG<sup>+</sup>18b,

[LT19, ABKW19, ABG19]. All transformations are generic. Both the works of [CDG+18b, ABKW19] need RO for their compilers. The transformation of [ABKW19, Section 4.2], however, does not support tags. The works [LT19] and [ABG19] independently provides compilers in the standard model that rely on PRF and symmetric-key encryption. The compiler in [LT19] is specified for MCFE and they rely on PRF with a special *multi-instance* security; The security model employed for DMCFE therein only constrains complete queries for the challenge ciphertexts and the DMCFE scheme of [LT19] achieves adaptive security under static corruption. The compiler in [ABG19] works also for DMCFE but their instantiation relies on single client FE with a mildly special property, namely *key derivation*. The DMCFE from [ABG19] achieves adaptive security under adaptive corruption. In the construction of *Dynamic Decentralized Functional Encryption* (DDFE) in [CDSG+20], of which DMCFE can also be seen as a particular case, the authors can achieve selective security against incomplete queries under static corruption using a primitive called *All-or-Nothing Encapsulation* (AoNE). All these works are for (D)MCFE in the security model firstly proposed in [CDG+18b]. We begin by recall the necessary building blocks for the transformation.

**All-or-Nothing Encapsulation (AoNE).** The notion of AoNE is a particular functionality of *Dynamic Decentralized Functional Encryption* (DDFE) introduced by Chotard *et al.* [CDSG+20]. In the transformation of [CDG+18b, Section 5.2], AoNE appears under the name *Secret Sharing Layer* (SSL). Let  $\mathcal{PK}$  denote a public key space. We recall the syntax of AoNE below:

- AoNE is a quadruple (AoNE.Setup, AoNE.KeyGen, AoNE.Enc, AoNE.Dec) defined over a key space  $\mathcal{K}$  and a message space  $\mathcal{M}$ .
- The procedure AoNE.Setup( $1^\lambda$ ) receives a security parameter and outputs the public parameters  $\text{pp}$ . The key space is defined as  $\mathcal{K} := \emptyset$  and the message space  $\mathcal{M} := \{0, 1\}^L \times \mathcal{PK}^n \times \text{Tag}$ , where  $L = L(\lambda)$  and  $n = n(\lambda)$  are functions. We denote by  $\text{List-PubKey}(n) \in \mathcal{PK}^n$  the list of  $n$  public keys for  $n$  senders, equipped with some ordering.
- The procedure AoNE.KeyGen(pp) outputs a public key  $\text{pk}$  and its associated secret key  $\text{sk}_{\text{pk}}$ .
- The procedure AoNE.Enc( $\text{sk}_{\text{pk}}, m$ ) parses  $m = (x_{\text{pk}}, \text{List-PubKey}(n), \text{tag}_{\text{aone}}) \in \mathcal{M}$ . Then, the procedure outputs  $(\text{pk}, m)$ .
- The procedure AoNE.Dec(ct) receives as input a list  $\text{ct} = (\text{ct}_{\text{pk}})_{\text{pk}}$  of ciphertexts indexed by public keys  $\text{pk} \in \mathcal{PK}$ . The procedure parses  $\text{ct}_{\text{pk}} := (\text{pk}, m_{\text{pk}})$  for each  $\text{ct}_{\text{pk}}$ . Finally, it outputs either  $(\text{pk}, x_{\text{pk}})_{\text{pk}}$  or  $\perp$ .

We recall in Appendix B.3 the definitions for *correctness* and *security* of AoNE, which will be needed for our transformation.

**The transformation.** Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKKeyComb}, \text{Enc}, \text{Dec})$  be a DMCFE, AoNE = (AoNE.Setup, AoNE.KeyGen, AoNE.Enc, AoNE.Dec) be an all-or-nothing encapsulation. We deduce a new DMCFE scheme  $\mathcal{E}^{(+)}$  as follows:

**Setup<sup>(+)</sup>( $1^\lambda$ ):** Execute  $((\text{sk}_i, \text{ek}_i)_{i=1}^n) \leftarrow \text{Setup}(1^\lambda)$ . Then, set up  $\text{pp} \leftarrow \text{AoNE.Setup}(1^\lambda)$  and for each  $i \in [n]$ , generate  $(\text{pk}_{\text{aone},i}^{\text{cip}}, \text{sk}_{\text{aone},i}^{\text{cip}})$  and  $(\text{pk}_{\text{aone},i}^{\text{key}}, \text{sk}_{\text{aone},i}^{\text{key}})$ , which are AoNE keys, using AoNE.KeyGen(pp). Output the *secret keys*  $\text{sk}_i$  and the *encryption keys*  $\text{ek}_i$  as follows

$$\begin{cases} \text{sk}_i^{(+)} & := (\text{sk}_i, \text{sk}_{\text{aone},i}^{\text{key}}) \\ \text{ek}_i^{(+)} & := (\text{ek}_i, \text{sk}_{\text{aone},i}^{\text{cip}}) \end{cases}.$$

We suppose that  $\text{List-PubKey}^{\text{cip}}(n) = (\text{ek}_{\text{aone},i}^{\text{cip}})_{i=1}^n$  and  $\text{List-PubKey}^{\text{key}}(n) = (\text{ek}_{\text{aone},i}^{\text{key}})_{i=1}^n$  are in the public parameters of the scheme.

**DKShare<sup>(+)</sup>( $\text{sk}_i^{(+)}, \text{tag-f}, F$ ):** Parse  $\text{sk}_i^{(+)} := (\text{sk}_i, \text{sk}_{\text{aone},i}^{\text{key}})$  and compute  $\text{dk}_{\text{tag-f},F,i} \leftarrow \text{DKShare}(\text{sk}_i, \text{tag-f}, F)$ . Then, compute

$$\text{dk}_{\text{tag-f},F,i}^{(+)} = \text{AoNE.Enc}(\text{sk}_{\text{aone},i}^{\text{key}}, (\text{dk}_{\text{tag-f},F,i}, \text{List-PubKey}^{\text{key}}(n), \text{tag-f})) .$$

Output  $\text{dk}_{\text{tag-f},F,i}^{(+)}$ .

$\text{DKeyComb}^{(+)}((\text{dk}_{\text{tag-f},F,i}^{(+)})_{i=1}^n, \text{tag-f}, F)$ : Run  $\text{AoNE.Dec}((\text{dk}_{\text{tag-f},F,i}^{(+)})_{i=1}^n) = (\text{dk}_{\text{tag-f},F,i})_{i=1}^n$ . Then, compute and output  $\text{dk}_{\text{tag-f},F} = \text{DKeyComb}((\text{dk}_{\text{tag-f},F,i})_{i=1}^n, \text{tag-f}, F)$ .

$\text{Enc}^{(+)}(\text{ek}_i^{(+)}, x_i, \text{tag})$ : Parse  $\text{ek}_i^{(+)} = (\text{ek}_i, \text{sk}_{\text{aone},i}^{\text{cip}})$  and compute  $\text{ct}_{\text{tag},i} \leftarrow \text{Enc}(\text{ek}_i, x_i, \text{tag})$ . Then, compute

$$\text{ct}_{\text{tag},i}^{(+)} = \text{AoNE.Enc}(\text{sk}_{\text{aone},i}^{\text{cip}}, (\text{ct}_{\text{tag},i}, \text{List-PubKey}^{\text{cip}}(n), \text{tag})) .$$

Output  $\text{ct}_{\text{tag},i}^{(+)}$ .

$\text{Dec}^{(+)}(\text{dk}_{\text{tag-f},F}, \mathbf{c}_{\text{tag}}^{(+)})$ : Parse  $\mathbf{c}_{\text{tag}}^{(+)} = (\text{ct}_{\text{tag},i}^{(+)})_{i=1}^n$ . Run  $\text{AoNE.Dec}((\text{ct}_{\text{tag},i}^{(+)})_{i=1}^n) = (\text{ct}_{\text{tag},i})_{i=1}^n$ . Finally, compute and output  $\text{Dec}(\text{dk}_{\text{tag-f},F}, (\text{ct}_{\text{tag},i})_{i=1}^n)$ .

*Correctness.* The correctness of  $\mathcal{E}^{(+)}$  follows that of AoNE and  $\mathcal{E}$ .

*Security.* The security of this transformation in the case of MCFE is studied in [CDG<sup>+</sup>18b, Theorem 12], where AoNE appears under the combination of *Secret Sharing Layer* (SSL) and *Symmetric Encryption* schemes with one-time security and can be proven secure under DBDH in the ROM. A concrete instantiation of AoNE following the same idea was presented in a recent work by Chotard *et al.* [CDSG<sup>+</sup>20, Section 5.2].

Back to the IND-security of  $\mathcal{E}^{(+)}$ , we can adapt the proofs in [CDSG<sup>+</sup>20, CDG<sup>+</sup>18b] to the case of DMCFE. In [CDG<sup>+</sup>18b, Theorem 12] for the MCFE case, the authors perform a sequence of hybrids on the challenge ciphertexts, which are indexed by  $\text{tag}$ , due to the fact that the AoNE is applied on the ciphertexts of the underlying MCFE scheme. In our case, we will have to perform in addition a similar sequence on the key components, which are indexed by  $\text{tag-f}$ , because we are treating DMCFE schemes instead of MCFE schemes, and this can be done following the same vein of [CDSG<sup>+</sup>20, Theorem 26]. Our new admissibility concerns only *complete* key and challenge queries thus the argument on the AoNE layer does not resort to admissibility. In summary, we can obtain the following:

**Theorem 24 (Adapted from [CDSG<sup>+</sup>20, CDG<sup>+</sup>18b]).** *Suppose  $\mathcal{E}$  be a DMCFE scheme that is IND-secure following Definition 7. Suppose AoNE is an all-or-nothing encapsulation scheme that is selectively statically IND-secure. Then, the DMCFE scheme  $\mathcal{E}^{(+)}$  is one-time IND-secure following Definition 18, selectively in challenge messages with static corruptions on both encryption and secret keys.*

### B.3 Correctness and Security of AoNE

*Correctness.* We require that for all  $x \in \{0, 1\}^L$  and  $\text{tag}_{\text{aone}} \in \text{Tag}$ , for all lists  $\text{List-PubKey}(n) \in \mathcal{PK}^n$ , we have  $\text{AoNE.Dec}((\text{pk}, (x_{\text{pk}}, \text{List-PubKey}(n), \text{tag}_{\text{aone}}))) = (\text{List-PubKey}(n), \text{tag}_{\text{aone}})$  and

$$\begin{aligned} & \text{AoNE.Dec}((\text{pk}, m_{\text{pk}})_{\text{pk} \in \text{List-PubKey}(n)}) \\ &= \begin{cases} (\text{pk}, x_{\text{pk}})_{\text{pk} \in \text{List-PubKey}(n)} \\ \quad \text{if } \forall \text{pk} \in \text{List-PubKey}(n) : m_{\text{pk}} = (x_{\text{pk}}, \text{List-PubKey}(n), \text{tag}_{\text{aone}}) \\ \perp \text{ otherwise} \end{cases} . \end{aligned}$$

*Security.* We state below the security notion of AoNE, proposed in [CDSG<sup>+</sup>20], that is adapted from DDFE where the former is an instance of the latter for a particular functionality.

**Definition 25 (IND-CPA security for AoNE).** *An all-or-nothing encapsulation scheme*

$$\text{AoNE} = (\text{AoNE.Setup}, \text{AoNE.KeyGen}, \text{AoNE.Enc}, \text{AoNE.Dec})$$

*for  $n$  senders over  $\{0, 1\}^L \times \mathcal{PK}^n \times \text{Tag}$  is IND-secure if for all ppt adversaries  $\mathcal{A}$ , and for all sufficiently large  $\lambda \in \mathbb{N}$ , the following probability is negligible*

$$\text{Adv}_{\text{AoNE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda) := \left| \Pr[\text{Expr}_{\text{AoNE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda) = 1] - \frac{1}{2} \right| .$$

<p><b>Initialise</b>(<math>1^\lambda</math>)</p> <p><math>b \xleftarrow{\\$} \{0, 1\}</math>  <math>\text{pp} \leftarrow \text{AoNE.Setup}(1^\lambda)</math>  <math>\mathcal{Q} := \emptyset, \mathcal{C} := \emptyset, \mathcal{H} := \emptyset</math>  Return <math>\text{pp}</math></p> <p><b>Enc</b>(<math>i, m, \text{pk}_i</math>)</p> <p>If there exists <math>(\text{pk}_i, \text{sk}_i)</math> in <math>\mathcal{Q}</math>  Return <math>\text{AoNE.Enc}(\text{sk}_i, m)</math>  Else: Ignore</p> <p><b>Finalise</b>(<math>b'</math>)</p> <p>If condition (*) holds:  return 0  Else: return <math>(b' \stackrel{?}{=} b)</math></p>	<p><b>LoR</b>(<math>i, m_i^{(0)}, m_i^{(1)}, \text{pk}_i</math>)</p> <p>If there exists <math>(\text{pk}_i, \text{sk}_i)</math> in <math>\mathcal{Q}</math>  Return <math>\text{AoNE.Enc}(\text{sk}_i, m_i^{(b)})</math>  Else: Ignore</p> <p><b>Corrupt</b>(<math>i, \text{pk}_i</math>)</p> <p>If there exists <math>(\text{pk}_i, \text{sk}_i)</math> in <math>\mathcal{Q}</math>  <math>\mathcal{C} := \mathcal{C} \cup \{i\}</math>  <math>\mathcal{H} := \mathcal{H} \setminus \{i\}</math>  Return <math>\text{sk}_i</math>  Else: Ignore</p> <p><b>NewHonest</b>(<math>i, \text{pp}</math>)</p> <p>If <math>i \in \mathcal{H}</math>: Ignore  <math>\mathcal{H} := \mathcal{H} \cup \{i\}</math>  <math>(\text{pk}_i, \text{sk}_i) \leftarrow \text{AoNE.KeyGen}(\text{pp}, i)</math>  <math>\mathcal{Q} := \mathcal{Q} \cup \{(\text{pk}_i, \text{sk}_i)\}</math>  Return <math>\text{pk}_i</math></p>
---	---

Fig. 2: The security games  $\text{Expr}_{\text{AoNE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda)$  for Definition 25. The condition (\*) in the game is defined in Definition 25.

The security game  $\text{Expr}_{\text{AoNE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda)$  is given in Figure 2.

Let the sets  $(\mathcal{C}, \mathcal{Q}, \mathcal{H})$  be the sets of corrupted clients, new participant queries, and honest clients, in that order. The condition (\*) used by  $\text{Expr}_{\text{AoNE}, \mathcal{A}}^{\text{ind-cpa}}(1^\lambda)$  is true if there exists a list of public key  $\text{List-PubKey}(n)$ , there exist two lists of messages  $\mathbf{m}^{(0)} = ((\text{pk}_i, m_i^{(0)}))_{\text{pk}_i}$  and  $\mathbf{m}^{(1)} = ((\text{pk}_i, m_i^{(1)}))_{\text{pk}_i}$  such that

$$\text{AoNE.Dec}(\mathbf{m}^{(0)}) \neq \text{AoNE.Dec}(\mathbf{m}^{(1)})$$

where for all  $i \in \mathcal{H}$ , there exists a query  $\text{LoR}(i, m_i^{(0)}, m_i^{(1)}, \text{pk}_i)$ . The probability is taken over the coins of algorithms and the choices of the adversary. Naturally, we can derive the selective notion  $\text{Adv}_{\text{AoNE}, \mathcal{A}}^{\text{sel-ind-cpa}}(1^\lambda)$  for the case where  $(\mathbf{m}^{(0)}, \mathbf{m}^{(1)})$  must be declared in advance, similarly the static notion  $\text{Adv}_{\text{AoNE}, \mathcal{A}}^{\text{stat-ind-cpa}}(1^\lambda)$  for which  $\mathcal{C}$  must be sent up front.

## C Supporting Materials - Deferred Proofs

### C.1 Proof of Lemma 8

**Lemma 8.** Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$  be a DMCFE scheme for the function class  $\mathcal{F}$ . If  $\mathcal{E}$  is one-time IND-secure, then  $\mathcal{E}$  is IND-secure.

*Proof.* Suppose  $\mathcal{E}$  is one-time IND-secure but not IND-secure. This means there exists a ppt adversary  $\mathcal{A}$  such that

$$\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda) := \left| \Pr[\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda) = 1] - \frac{1}{2} \right|$$

is non-negligible. We construct a ppt adversary  $\mathcal{B}$  using a black-box access to  $\mathcal{A}$  to break the one-time IND-security of  $\mathcal{E}$ . We note that this reduction preserves the incomplete/complete constraint, that is, we restrict  $\mathcal{B}$  to query all honest components if and only if we restrict  $\mathcal{A}$  to do the same. The adversary  $\mathcal{B}$  works as follows:

1.  $\mathcal{B}$  first obtains  $\text{pk}$  from its one-time challenger and sends  $\text{pk}$  to  $\mathcal{A}$ .
2.  $\mathcal{B}$  chooses uniformly at random  $k \in [Q]$ .
3. For all challenge ciphertext queries  $\text{LoR}(i, x_{k,i}^{(0)}, x_{k,i}^{(1)}, \text{tag}_k^*)$ :  $\mathcal{B}$  make the queries to  $\text{LoR}$  to its one-time challenger for  $\text{tag}_k^*$  and transfer the responses to  $\mathcal{A}$ .
4. For all challenge ciphertext queries  $\text{LoR}(i, x_{\ell,i}^{(0)}, x_{\ell,i}^{(1)}, \text{tag}_\ell^* \neq \text{tag}_k^*)$  where  $\ell < k$ :  $\mathcal{B}$  makes the queries for  $x_{\ell,i}^{(0)}$  to  $\text{Enc}$  to its one-time challenger and transfer the responses to  $\mathcal{A}$ .
5. For all challenge ciphertext queries  $\text{LoR}(i, x_{\ell,i}^{(0)}, x_{\ell,i}^{(1)}, \text{tag}_\ell^* \neq \text{tag}_k^*)$  where  $\ell > k$ :  $\mathcal{B}$  makes the queries for  $x_{\ell,i}^{(1)}$  to  $\text{Enc}$  of its one-time challenger and transfer the responses to  $\mathcal{A}$ .
6. For all  $\text{Enc}$  and  $\text{DKeyGenShare}$  queries by  $\mathcal{A}$ ,  $\mathcal{B}$  relay them to its one-time challenger and transfers the responses to  $\mathcal{A}$ .
7. Finally,  $\mathcal{A}$  outputs a bit  $b'$ . The adversary  $\mathcal{B}$  outputs the same bit  $b'$ .

Let  $Q$  denote the number of challenge tags that are queried by  $\mathcal{A}$  to  $\mathcal{B}$ . We use a sequence of hybrids  $\mathcal{H}_0, \mathcal{H}_1, \dots, \mathcal{H}_Q$  between  $\mathcal{A}$  and its IND-security challenger such that in  $\mathcal{H}_k$  all  $\text{LoR}$  queries for  $\text{tag}_\ell^*$ , where  $\ell \leq k$ , are answered by encrypting  $x_{\ell,i}^{(0)}$  and by encrypting  $x_{\ell,i}^{(1)}$  if  $\ell > k$ .

We denote by  $\mathcal{H}_k = 1$  the event where the challenger outputs 1. We have  $2 \cdot \text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda) = |\Pr[\mathcal{H}_Q = 1] - \Pr[\mathcal{H}_0 = 1]|$ .

On the other hand, the advantage of  $\mathcal{B}$  against the one-time IND-security experiment of  $\mathcal{E}$  is

$$\begin{aligned}
2 \cdot \text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda) &= |\Pr[\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda) = 1 \mid b = 0] \\
&\quad - \Pr[\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda) = 1 \mid b = 1]| \\
&= \frac{1}{Q} \cdot \left| \sum_{k=1}^Q \left( \Pr[\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda) = 1 \mid b = 0, \mathcal{B} \text{ picks } k] \right. \right. \\
&\quad \left. \left. - \Pr[\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda) = 1 \mid b = 1, \mathcal{B} \text{ picks } k] \right) \right| \\
&\stackrel{(*)}{\geq} \frac{1}{Q} \cdot \left| \sum_{k=1}^Q (\Pr[\mathcal{H}_k = 1] - \Pr[\mathcal{H}_{k-1} = 1]) \right| \\
&= \frac{1}{Q} \cdot |\Pr[\mathcal{H}_Q = 1] - \Pr[\mathcal{H}_0 = 1]| \\
&= \frac{1}{Q} \cdot 2 \cdot \text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda)
\end{aligned}$$

where  $(*)$  comes from the observation that conditioned on  $b = 0$  (resp.  $b = 1$ ) and  $\mathcal{B}$  picks  $k$ ,  $\text{Expr}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda) = 1$  is identical to  $\mathcal{H}_k = 1$  (resp.  $\mathcal{H}_{k-1} = 1$ ), where  $\mathcal{B}$  is simulating the challenger for  $\mathcal{A}$ . Finally, we have  $\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda) \leq Q \cdot \text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{B}}^{\text{dmc-ind-cpa-1chal}}(1^\lambda)$  and because  $\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-ind-cpa}}(1^\lambda)$  is non-negligible,  $\mathcal{B}$  is breaking the one-time IND-security of  $\mathcal{E}$  with non-negligible advantage.  $\square$

## C.2 Proof of Theorem 16

**Theorem 16.** *Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$  be a DMCFE scheme for  $\mathcal{F}^{\text{IP}}$  from Section 5.1 in a bilinear group  $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ . Then,  $\mathcal{E}$  is IND-secure with static corruption of secret keys in the ROM if the SXDH assumption holds for  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . More specifically, let  $n$  denote the dimension for inner-products and  $Q_1, Q_2$  denote the maximum number of random oracle (RO) queries to  $\mathbf{H}_1, \mathbf{H}_2$  respectively. For any ppt adversary  $\mathcal{A}$  against  $\mathcal{E}$  with static secret key corruption and one-time challenge, we have the following bound:*

$$\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-stat-sk-ind-cpa-1chal}}(1^\lambda) \leq (3 + 2Q_1) \cdot \text{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\text{SXDH}}(1^\lambda) + \frac{Q_2^2}{2q}$$



and in the reduction there is an additive loss  $\mathcal{O}(Q_1 \cdot t_{\mathbb{G}_1})$  in time, where  $t_{\mathbb{G}_1}$  is the cost for one addition in  $\mathbb{G}_1$ .

*Proof (Of Theorem 16).* We give the sequence of games in Figure 3 and Figure 4. The changes that make the transitions between games are highlighted in gray. The advantage of an adversary  $\mathcal{A}$  in a game  $\mathbb{G}_i$  is denoted by

$$\text{Adv}(\mathbb{G}_i) := |\Pr[\mathbb{G}_i = 1] - 1/2|$$

where the probability is taken over the random choices of  $\mathcal{A}$  and coins of  $\mathbb{G}_i$ . The full domain hash function  $\mathbf{H}_1 : \text{Tag} \rightarrow \mathbb{G}_1^2$  as well as the hash function  $\mathbf{H}_1 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$  are modeled as random oracles (RO). We denote by  $K$  the number of key queries, in the proof we omit the indexing of keys for readability. We consider only admissible adversaries and recall that the following must hold:

$$\left\{ \begin{array}{l} \sum_{i \in \mathcal{H}} \Delta \mathbf{x}[i] \mathbf{y}[i] = 0 \text{ where } \mathcal{H} := \mathcal{H}_{\text{ekey}} \cup \mathcal{H}_{\text{skey}}, \\ i^* \in \mathcal{C}_{\text{ekey}} : \text{either } \mathbf{x}^{(0)}[i^*] = \mathbf{x}^{(1)}[i^*] \text{ or } \mathbf{y}[i^*] = 0, \\ i^* \in \mathcal{C}_{\text{skey}} : \mathbf{x}^{(0)}[i^*] = \mathbf{x}^{(1)}[i^*] . \end{array} \right. \quad (7)$$

**Game  $\mathbb{G}_0$ :** This is the security game  $\text{Exp}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-stat-sk-ind-cpa-1chal}}(1^\lambda)$ . The order of ciphertext or key queries, for which the adversary is supposed to get the whole  $n$  components, and the corruption the clients over  $\text{ek}_i$  or  $\text{sk}_i$  can be arbitrary. The adversary will declare the sets  $\mathcal{C}_{\text{skey}}$  of corrupted secret keys  $\text{sk}_i$  at the beginning of the game. We have  $\text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-stat-sk-ind-cpa-1chal}}(1^\lambda) = \text{Adv}(\mathbb{G}_0)$ .

**Game  $\mathbb{G}_1$ :** In this game we introduce a random multiple  $r_i \tau$  of  $r_i$  in the 5-th coordinate of all  $\mathbf{c}_{i, \text{ipfe}}$  returned from **LoR**. The basis changes are performed at **Setup** for all  $i$ . This basis change is computational and we will see below that we can write appropriately the vectors from **LoR** so as to introduce  $r_i \tau$ , while those from **Enc** stay normal. Given a DDH instance  $(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket c \rrbracket_1)$ , where  $c - ab = \tau$  is 0 or a random value in  $\mathbb{Z}_q$ , the basis change uses the matrices:

$$\begin{aligned} H_i &:= \begin{bmatrix} 1 & a_i \\ 0 & 1 \end{bmatrix}_{3,5} & H'_i &:= (H_i^{-1})^\top = \begin{bmatrix} 1 & 0 \\ -a_i & 1 \end{bmatrix}_{3,5} \\ \mathbf{H}_i &= H_i \cdot \mathbf{T}; & \mathbf{H}_i^* &= H'_i \cdot \mathbf{T}^* . \end{aligned}$$

Here we are using the random self-reducibility of DDH to obtain instances  $(\llbracket a_i \rrbracket_1, \llbracket b_i \rrbracket_1, \llbracket c_i \rrbracket_1)$  for  $c_i - a_i b_i = \tau_i$  is 0 or a uniformly random value in  $\mathbb{Z}_q$ . The vectors from **LoR** for the corrupted  $i$ , if demanded by the adversary, can be written:

$$\mathbf{c}_{i, \text{ipfe}} = (r'_i t_i b_i, r_i t_i c_i)_{\mathbf{T}[3,5]} = (r'_i t_i b_i, r'_i t_i \tau_i)_{\mathbf{H}_i[3,5]}$$

and the key components  $\mathbf{k}_{i, \text{ipfe}}$  stay unchanged and we do not have problems for simulating them:

$$\mathbf{k}_{i, \text{ipfe}} = (v_i y_i / t_i, 0)_{\mathbf{T}[3,5]} = (v_i y_i / t_i, 0)_{\mathbf{H}_i[3,5]} .$$

We are simulating  $r_i := r'_i b_i$  and  $t_i \tau_i$  plays the role of the random factor to  $r'_i$ . The basis change affects  $\mathbf{h}_{i,3}$  and  $\mathbf{h}_{i,5}^*$ . We do not have  $\llbracket a \rrbracket_2$  to compute  $\mathbf{h}_{i,5}^*$  but currently its coefficient is 0 is all vectors thus there are no problems. However, since  $\mathbf{h}_{i,3}$  appears only in a random linear combination in  $\text{ek}_i$  and  $\mathbf{h}_{i,5}^*$  is in the *hidden* part of the basis, the change cannot be recognized via the corrupted  $\text{ek}_i$  and the ciphertexts from **Enc** can still be simulated. The correctness of decryption w.r.t the challenge ciphertext **LoR** is preserved. We have  $|\text{Adv}(\mathbb{G}_1) - \text{Adv}(\mathbb{G}_0)| \leq \text{Adv}_{\mathbb{G}_1}^{\text{DDH}}(1^\lambda)$ .

**Game G<sub>0</sub>** :  $\mathbf{H}_1(\text{tag}) \rightarrow (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1), \mathbf{H}_1(\text{tag}') \rightarrow (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \chi'_{\text{tag}'} \rrbracket_1), \zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma'_i, \theta \xleftarrow{\$} \mathbb{Z}_q$   
 where  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4; \sum_{i \in [n]} \theta_i = 0; \mathcal{C}_{\text{sk}_i}$  is selectively declared

$$\forall i \in [n] : \mathbf{ek}_i (p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)}, q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)})$$

$$\forall i \in [n] : \mathbf{sk}_i (s_i \alpha_i \mathbf{h}_{i,1}^* + s_i \gamma_i \mathbf{h}_{i,2}^*, u_i \gamma'_i \mathbf{h}_{i,1}^* + u_i \alpha_i \mathbf{h}_{i,2}^*, \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*, \theta_i \mathbf{h}_{i,6}^*)$$

**LoR**

$$(\omega_{\text{tag}} p_i \mid \omega'_{\text{tag}} q_i \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i \mid 0 \mid \omega_{\text{tag}})_{\mathbf{H}_i}$$

**Enc**

$$(\chi_{\text{tag}'} p_i \mid \chi'_{\text{tag}'} q_i \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \chi'_{\text{tag}'} \zeta_2) \cdot s_i - (\chi'_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'})_{\mathbf{H}_i}$$

**DKeyGenShare**

$$(s_i y_i \alpha_i + u_i y_i \gamma'_i \mid s_i y_i \gamma_i + u_i y_i \alpha_i \mid \frac{v_i}{t_i} y_i \mid y_i \mid 0 \mid d_{\text{tag-f,y}} \theta_i)_{\mathbf{H}_i^*}$$

**Game G<sub>1</sub>** :  $\mathbf{H}_1(\text{tag}) \rightarrow (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1), \mathbf{H}_1(\text{tag}') \rightarrow (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \chi'_{\text{tag}'} \rrbracket_1), \zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma'_i, \theta \xleftarrow{\$} \mathbb{Z}_q$   
 where  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4; \sum_{i \in [n]} \theta_i = 0; \mathcal{C}_{\text{sk}_i}$  is selectively declared;  $r'_i, \tau_i \xleftarrow{\$} \mathbb{Z}_q$

**LoR**

$$(\cdots \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - r'_i \tau_i \mid \omega_{\text{tag}})_{\mathbf{H}_i}$$

**Enc**

$$(\cdots \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \chi'_{\text{tag}'} \zeta_2) \cdot s_i - (\chi'_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'})_{\mathbf{H}_i}$$

**DKeyGenShare**

$$(\cdots \mid \frac{v_i}{t_i} y_i \mid y_i \mid 0 \mid d_{\text{tag-f,y}} \theta_i)_{\mathbf{H}_i^*}$$

**Game G<sub>2</sub>** :  $\mathbf{H}_1(\text{tag}) \rightarrow (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1), \mathbf{H}_1(\text{tag}') \rightarrow (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \chi'_{\text{tag}'} \rrbracket_1), \zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma'_i, \theta \xleftarrow{\$} \mathbb{Z}_q$   
 where  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4; \sum_{i \in [n]} \theta_i = 0; \mathcal{C}_{\text{sk}_i}$  is selectively declared;  $\tau_i, \rho \xleftarrow{\$} \mathbb{Z}_q$

**LoR**

$$(\cdots \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - r'_i \rho \tau_i \mid r'_i \tau_i \mid \omega_{\text{tag}})_{\mathbf{H}_i}$$

**Enc**

$$(\cdots \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \chi'_{\text{tag}'} \zeta_2) \cdot s_i - (\chi'_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'})_{\mathbf{H}_i}$$

**DKeyGenShare**

$$(\cdots \mid \frac{v_i}{t_i} y_i \mid y_i \mid \rho y_i \mid d_{\text{tag-f,y}} \theta_i)_{\mathbf{H}_i^*}$$

**Game G<sub>3</sub>** :  $\mathbf{H}_1(\text{tag}) \rightarrow (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1), \mathbf{H}_1(\text{tag}') \rightarrow (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \chi'_{\text{tag}'} \rrbracket_1), \zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma'_i, \theta \xleftarrow{\$} \mathbb{Z}_q$   
 where  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4; \sum_{i \in [n]} \theta_i = 0; \mathcal{C}_{\text{sk}_i}$  is selectively declared;  $\tau_i, \rho \xleftarrow{\$} \mathbb{Z}_q$

**LoR**

$$(\cdots \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - r'_i \rho \tau_i \mid r'_i \rho \tau_i \mid \omega_{\text{tag}})_{\mathbf{H}_i}$$

**Enc**

$$(\cdots \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \chi'_{\text{tag}'} \zeta_2) \cdot s_i - (\chi'_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'})_{\mathbf{H}_i}$$

**DKeyGenShare**

$$(\cdots \mid \frac{v_i}{t_i} y_i \mid y_i \mid y_i \mid d_{\text{tag-f,y}} \theta_i)_{\mathbf{H}_i^*}$$

Fig. 3: Games  $\mathbf{G}_0, \dots, \mathbf{G}_3$  for Theorem 16. The set  $\mathcal{C}_{\text{sk}_i}$  of corrupted secret keys  $\mathbf{sk}_i$  is selectively declared. The index  $\ell$  runs in  $\{1, \dots, K\}$  for the functional key queries and  $k$  runs in the number of ciphertexts (including  $n$  components) queried to **Enc**. The function  $\mathbf{H}_1$  is modeled as a random oracle.

**Game G<sub>2</sub>**: In this game, we perform a formal basis change to introduce a random value in the 4-th coordinate of  $\mathbf{c}_{i,\text{ipfe}}$  and a multiple of  $y_i$  in the 5-th coordinate of  $\mathbf{k}_{i,\text{ipfe}}$  for *all* keys. The basis changes are performed at **Setup** for  $i \in \mathcal{H}_{\text{sk}_i}$  statically known from the beginning. The simulator samples  $\rho \xleftarrow{\$} \mathbb{Z}_q$  and uses the matrices:

$$\begin{aligned} H_i &:= \begin{bmatrix} 1 & 0 \\ \rho & 1 \end{bmatrix}_{4,5} & H'_i &:= (H_i^{-1})^\top = \begin{bmatrix} 1 & -\rho \\ 0 & 1 \end{bmatrix}_{4,5} \\ \mathbf{H}_i &= H_i \cdot \mathbf{T}; & \mathbf{H}_i^* &= H'_i \cdot \mathbf{T}^* . \end{aligned}$$

The vectors from **LoR** for the corrupted  $i$  become:

$$\begin{aligned} \mathbf{c}_{i,\text{ipfe}} &= (-(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i, r'_i \tau_i) \mathbf{T}[4,5] \\ &= (-(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - r'_i \rho \tau_i, r'_i \tau_i)_{\mathbf{H}_i}[4,5] \end{aligned}$$

**Game G<sub>4</sub>** :  $H_1(\text{tag}) \rightarrow (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1)$ ,  $H_1(\text{tag}') \rightarrow (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \chi'_{\text{tag}'} \rrbracket_1)$ ,  $\zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma'_i, \theta \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  where  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4; \sum_{i \in [n]} \theta_i = 0; \mathcal{C}_{\text{skey}}$  is selectively declared;  $\tilde{\rho}_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$

**LoR**  
 $(\cdots \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - \tilde{\rho}_i \tau_i \mid \tilde{\rho}_i \tau_i \mid \omega_{\text{tag}} \mid)_{H_i}$   
**Enc**  
 $(\cdots \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \chi'_{\text{tag}'} \zeta_2) \cdot s_i - (\chi'_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'} \mid)_{H_i}$   
**DKeyGenShare**  
 $(\cdots \mid \frac{v_i}{t_i} y_i \mid y_i \mid y_i \mid d_{\text{tag-f}, y} \theta_i)_{H_i^*}$

**Game G<sub>5</sub>** :  $H_1(\text{tag}) = \llbracket \text{RF}(\text{tag}) \rrbracket_1 := (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1)$ ,  $H_1(\text{tag}') = \llbracket \text{RF}(\text{tag}') \rrbracket_1 := (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \chi'_{\text{tag}'} \rrbracket_1)$ ,  
 $H_2(\text{tag-f}, y)_2 = \llbracket \text{RF}_2(\text{tag-f}, y) \rrbracket_1 := d_{\text{tag-f}, y}$

**Game G<sub>6</sub>** :  $\mu \stackrel{\$}{\leftarrow} \mathbb{Z}_q, H_1(\text{tag}) := \llbracket \text{RF}(\text{tag}) \rrbracket_1 := (\llbracket \omega_{\text{tag}} \rrbracket_1, \llbracket \omega'_{\text{tag}} \rrbracket_1)$ ,  $H_1(\text{tag}') := \llbracket \text{RF}'(\text{tag}') \cdot (1, \mu) \rrbracket_1 = (\llbracket \chi_{\text{tag}'} \rrbracket_1, \llbracket \mu \chi_{\text{tag}'} \rrbracket_1)$

**LoR**  
 $(\cdots \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - \tilde{\rho}_i \tau_i \mid \tilde{\rho}_i \tau_i \mid \omega_{\text{tag}} \mid)_{H_i}$   
**Enc**  
 $(\cdots \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \mu \chi_{\text{tag}'} \zeta_2) \cdot s_i - (\mu \chi_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'} \mid)_{H_i}$   
**DKeyGenShare**  
 $(\cdots \mid \frac{v_i}{t_i} y_i \mid y_i \mid y_i \mid d_{\text{tag-f}, y} \theta_i)_{H_i^*}$

**Game G<sub>7</sub>** :  $d_{\text{tag-f}, y, i} \stackrel{\$}{\leftarrow} \mathbb{Z}_q, \sum_{i=1}^n d_{\text{tag-f}, y, i} = 0$

**LoR**  
 $(\cdots \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \cdot s_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \cdot u_i + \mathbf{x}_b^*[i] - r_i v_i - \tilde{\rho}_i \tau_i \mid \tilde{\rho}_i \tau_i \mid \omega_{\text{tag}} \mid)_{H_i}$   
**Enc**  
 $(\cdots \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \mu \chi_{\text{tag}'} \zeta_2) \cdot s_i - (\mu \chi_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \cdot u_i + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'} \mid)_{H_i}$   
**DKeyGenShare**  
 $(\cdots \mid \frac{v_i}{t_i} y_i \mid y_i \mid y_i \mid d_{\text{tag-f}, y, i})_{H_i^*}$

**Game G<sub>8</sub>** :  $\zeta_1, \zeta_2, \zeta_3, \zeta_4, p_i, q_i, \alpha_i, \gamma_i, \gamma'_i, \theta_i, \theta'_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  where  $p_i \alpha_i = \zeta_1, q_i \gamma_i = \zeta_2, q_i \alpha_i = \zeta_3, p_i \gamma'_i = \zeta_4$  and  $\sum_{i=1}^n \theta_i = \sum_{i=1}^n \theta'_i = 0$ ; Define  $S' = S + \Delta S, U' = U + \Delta U$ , where  $\Delta S, \Delta U \in \mathbb{Z}_q^n$  s.t. for each  $i$ :

$$\begin{cases} (\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \Delta S[i] + (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \Delta U[i] = -(\mathbf{x}_b^*[i] - \mathbf{x}_0^*[i]) \\ \Delta S[i](\alpha_i + \gamma_i \frac{\zeta_3}{\zeta_1} \mu) + \Delta U[i](\gamma'_i + \alpha_i \frac{\zeta_3}{\zeta_1} \mu) = 0 \end{cases}$$

$$\forall i \in [n] : \text{sk}_i (\mathbf{s}_i^* \alpha_i \mathbf{h}_{i,1}^* + \mathbf{s}_i^* \gamma_i \mathbf{h}_{i,2}^* + \mathbf{u}_i^* \gamma'_i \mathbf{h}_{i,1}^* + \mathbf{u}_i^* \alpha_i \mathbf{h}_{i,2}^* - \frac{v_i}{t_i} \mathbf{h}_{i,3}^* + \mathbf{h}_{i,4}^*, \mathbf{h}_{i,6}^*)$$

**LoR**  
 $(\omega_{\text{tag}} p_i \mid \omega'_{\text{tag}} q_i \mid r_i t_i \mid -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) \mathbf{s}_i^* - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) \mathbf{u}_i^* + \mathbf{x}_0^*[i] - r_i v_i - \tilde{\rho}_i \tau_i \mid \tilde{\rho}_i \tau_i \mid \omega_{\text{tag}} \mid)_{H_i}$   
**Enc**  
 $(\chi_{\text{tag}'} p_i \mid \mu \chi_{\text{tag}'} q_i \mid r_i^{(k)} t_i \mid -(\chi_{\text{tag}'} \zeta_1 + \mu \chi_{\text{tag}'} \zeta_2) \mathbf{s}_i^* - (\mu \chi_{\text{tag}'} \zeta_3 + \chi_{\text{tag}'} \zeta_4) \mathbf{u}_i^* + x_i - r_i^{(k)} v_i \mid 0 \mid \chi_{\text{tag}'} \mid)_{H_i}$   
**DKeyGenShare**  
 $((\mathbf{s}_i^* \alpha_i + \mathbf{u}_i^* \gamma'_i) y_i \mid (\mathbf{s}_i^* \gamma_i + \mathbf{u}_i^* \alpha_i) y_i \mid \frac{v_i}{t_i} y_i \mid y_i \mid d_{\text{tag-f}, y, i})_{H_i^*}$

Fig. 4: Games  $G_4, \dots, G_8$  for Theorem 16. The set  $\mathcal{C}_{\text{skey}}$  of corrupted secret keys  $\text{sk}_i$  is selectively declared. The index  $\ell$  runs in  $\{1, \dots, K\}$  for the functional key queries and  $k$  runs in the number of ciphertexts (including  $n$  components) queried to **Enc**. In  $G_5$  we use random functions  $\text{RF} : \text{Tag} \rightarrow (\mathbb{Z}_q^*)^2$ ,  $\text{RF}_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ . In  $G_6$  we use a random function  $\text{RF}' : \text{Tag} \rightarrow \mathbb{Z}_q^*$ .

and the key components  $\mathbf{k}_{i, \text{ipfe}}$  become

$$\mathbf{k}_{i, \text{ipfe}} = (y_i, 0)_{\mathbf{T}^*[4,5]} = (y_i, y_i \rho)_{H_i^*[4,5]}.$$

The basis change affects  $\mathbf{h}_{i,5}$  and  $\mathbf{h}_{i,4}^*$ . However, since  $\mathbf{h}_{i,4}^*$  appears only in a random linear combination in  $\text{sk}_i$ , among which the corrupted we know in advance, and  $\mathbf{h}_{i,5}$  is in the *hidden* part of the basis, the change is perfectly indistinguishable. The correctness of decryption w.r.t the challenge ciphertext **LoR** is preserved. We do not modify the ciphertexts from **Enc**, because their 5-th coordinates are 0 from previous games, and the normal ciphertexts from **Enc** retain the correctness. Moreover, because it is a formal basis change at **Setup**, all the key components  $\mathbf{k}_{i, \text{ipfe}}$  will be affected, no matter they are computed by **DKeyGenShare**

or by a corrupted  $\mathbf{sk}_i$  and will have the desired modification. In the end it holds that  $\text{Adv}(\mathbb{G}_2) = \text{Adv}(\mathbb{G}_1)$ .

**Game  $\mathbb{G}_3$ :** We perform a formal basis change on the 5-th coordinate of  $(\mathbf{H}_i, \mathbf{H}_i^*)$  to “move”  $\rho$ . The basis changes are performed at **Setup**, which is known due to the static corruption constraint. The simulator uses the matrices:

$$\begin{aligned} H_i &:= [\rho]_5 & H'_i &:= (H_i^{-1})^\top = \left[\frac{1}{\rho}\right]_5 \\ \mathbf{H}_i &= H_i \cdot \mathbf{T}; & \mathbf{H}_i^* &= H'_i \cdot \mathbf{T}^* . \end{aligned}$$

The change is performed on the hidden part, preserving the correctness and thus  $\text{Adv}(\mathbb{G}_3) = \text{Adv}(\mathbb{G}_2)$ . Similar to the transition from  $\mathbb{G}_1$  to  $\mathbb{G}_2$ , regardless of whether  $\mathbf{k}_{i,\text{ipfe}}$  is written by **DKeyGenShare** or by a corrupted  $\mathbf{sk}_i$ , this formal basis change still has its affect and the vectors are modified as required.

**Game  $\mathbb{G}_4$ :** We use the DDH assumption in  $\mathbb{G}_1$  to change  $r'_i \rho$  from  $\mathbb{G}_3$  into a totally random value  $\tilde{\rho}_i$ . We recall that all the changes thus far are assured for any query  $i$  to **LoR**. Hence, given a DDH instance  $(\llbracket r'_i \rrbracket_1, \llbracket \rho \rrbracket_1, \llbracket \tilde{\rho}_i \rrbracket_1)$ , we can simulate these responses. If  $\tilde{\rho}_i = r'_i \rho$  then the responses follow  $\mathbb{G}_3$ , else they follow  $\mathbb{G}_4$ . We have  $|\text{Adv}(\mathbb{G}_4) - \text{Adv}(\mathbb{G}_3)| \leq \text{Adv}_{\mathbb{G}_1}^{\text{DDH}}(1^\lambda)$ , by using the random self-reducibility of DDH to obtain  $n$  DDH instances.

**Game  $\mathbb{G}_5$ :** In  $\mathbb{G}_5$  we use a random function  $\text{RF} : \text{Tag} \rightarrow (\mathbb{Z}_q^*)^2$  to simulate the RO of  $\mathbf{H}_1$ . The distribution of  $\text{RF}(\text{tag}) \rightarrow \llbracket (\omega_{\text{tag}}, \omega'_{\text{tag}}) \rrbracket_1$  is identical to that from  $\mathbf{H}_1$ . Moreover, we also simulate the RO of  $\mathbf{H}_2$  by  $\text{RF}_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ . The distribution of  $\text{RF}_2(\text{tag-f}, \mathbf{y})$  stays the same, except that the simulator aborts in **Finalize** if there is a collision among  $Q_2$  queries to  $\mathbf{H}_2$ . The collision probability is bounded by  $\frac{Q_2^2}{2q}$ , where it is taken over the uniformly randomly independent choices for each of the  $Q_2$  queries. In the end, we have  $|\text{Adv}(\mathbb{G}_5) - \text{Adv}(\mathbb{G}_4)| \leq \frac{Q_2^2}{2q}$ .

**Game  $\mathbb{G}_6$ :** In this game we replace the shifted secret shares of 0 in  $\mathbf{k}_{i,\text{ipfe}}$ , which are  $d_{\text{tag-f}, \mathbf{y}} \theta_i$ , while we simulate the RO of  $\mathbf{H}_2$  by  $\text{RF}_2 : \text{Tag} \times \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$  and  $(\theta_i)_{i=1}^n$  is a random secret sharing of 0 generated from **Setup**. Our goal is to replace  $(d_{\text{tag-f}, \mathbf{y}} \theta_i)_{i=1}^n$  by  $(d_{\text{tag-f}, \mathbf{y}, i})_{i=1}^n$  that are independent secret sharings of 0.

Our key observation is that: because we are in the *static* corruption model for **skey**, all *corrupted*  $i$  are known since the beginning. More specifically, the secret shares  $(\theta_i)_{i=1}^n$  are generated at setup and  $\sum_{i \in \mathcal{H}_{\text{skey}}} \theta_i = -\left(\sum_{i \in \mathcal{C}_{\text{skey}}} \theta_i\right)$  is fixed since the beginning. Therefore, upon receiving the tag **tag-f** and  $\mathbf{y}$  the sum:

$$R_{\mathbf{y}} := d_{\text{tag-f}, \mathbf{y}} \sum_{i \in \mathcal{H}_{\text{skey}}} \theta_i$$

is fixed in advance. We use this observation and the *random-self reducibility* of DDH in  $\mathbb{G}_2$  in a sequence of hybrids  $\mathbb{G}_{5,k}$  over  $k \in [K] \cup \{0\}$ , where  $K$  is the number of key queries, for changing the key component  $\mathbf{k}_{i,\text{ipfe}}$  under **tag-f**.

In the hybrid  $\mathbb{G}_{5,k}$  with  $0 \leq k \leq K + 1$ , the first  $k$  key queries  $\mathbf{k}_{i,\text{ipfe}}$  are having a random secret shares over  $i \in \mathcal{H}$ :

$$\mathbf{k}_{i,\text{ipfe}} = \left( \cdots, \frac{v_i}{t_i} y_i, y_i, y_i, d_{\text{tag-f}, \mathbf{y}, i} \right) \mathbf{H}_i^*$$

where  $d_{\text{tag-f}, \mathbf{y}, i} \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  and  $\sum_{i \in \mathcal{H}} d_{\text{tag-f}, \mathbf{y}, i} = R_{\mathbf{y}} = d_{\text{tag-f}, \mathbf{y}} \sum_{i \in \mathcal{H}_{\text{skey}}} \theta_i$ . We have  $\mathbb{G}_{5,0} = \mathbb{G}_5$  and  $\mathbb{G}_{5,K+1} = \mathbb{G}_6$ .

We describe the transition from  $\mathbb{G}_{5,k-1}$  to  $\mathbb{G}_{5,k}$  for  $k \in [K + 1]$ , using a DDH instance  $(\llbracket a \rrbracket_2, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2)$  where  $c - ab = 0$  or a uniformly random value. Given a ppt adversary  $\mathcal{A}$  that can distinguish  $\mathbb{G}_{5,k-1}$  from  $\mathbb{G}_{5,k}$  that differ at the  $k$ -th key query, we build a ppt adversary  $\mathcal{B}$  that breaks the DDH in  $\mathbb{G}_2$ :

- The adversary  $\mathcal{B}$  uses  $\llbracket a \rrbracket_2$  to simulate  $\mathbf{H}_2(\text{tag-f}, \mathbf{y})$ . This implicitly sets  $d_{\text{tag-f}, \mathbf{y}} := a$ .

- The adversary  $\mathcal{B}$  samples  $\theta_i \xleftarrow{\$} \mathbb{Z}_q$  for corrupted  $i \in \mathcal{C}_{\text{sk}_e}$ , as well as other parameters to output the corrupted keys  $\text{sk}_i$  to  $\mathcal{A}$ . Then,  $\mathcal{B}$  computes and defines  $S_k := -\sum_{i \in \mathcal{C}_{\text{sk}_e}} \theta_i$ .
- Let us denote  $H := |\mathcal{H}_{\text{sk}_e}|$  the number of honest  $i$ , For  $i$  among the first  $H - 1$  honest clients whose keys are never leaked,  $\mathcal{B}$  uses the *random-self reducibility* to compute  $\llbracket d_{\text{tag-f},y} \theta_i \rrbracket_2$  for responding to the  $k$ -th key query  $\mathbf{k}_{i,\text{ipfe}}$ .
- First of all, for  $i$  among the first  $|\mathcal{H}| - 1$  honest,  $\mathcal{B}$  samples  $\alpha_{k,i}, \beta_{k,i} \xleftarrow{\$} \mathbb{Z}_q$  and implicitly defines  $b_{k,i} := \alpha_{k,i}b + \beta_{k,i}$ ,  $c_{k,i} := \alpha_{k,i}c + \beta_{k,i}a$ . We note that

$$\begin{cases} \llbracket b_{k,i} \rrbracket_2 &= \alpha_{k,i} \llbracket b \rrbracket_2 + \llbracket \beta_{k,i} \rrbracket_2 \\ \llbracket c_{k,i} \rrbracket_2 &= \alpha_{k,i} \llbracket c \rrbracket_2 + \beta_{k,i} \llbracket a \rrbracket_2 \end{cases}$$

are efficiently computable from the DDH instance. Then,  $\mathcal{B}$  uses  $\llbracket c_{k,i} \rrbracket_2$  in the simulation of  $\mathbf{k}_{i,\text{ipfe}}$ .

- Next, for the last  $H$ -th honest client,  $\mathcal{B}$  computes and defines:

$$\llbracket c_{k,H} \rrbracket_2 := S_k \cdot \llbracket a \rrbracket_2 - \sum_{i \in \mathcal{H}_{\text{sk}_e} \setminus \{H\}} \llbracket c_{k,i} \rrbracket_2 \quad (8)$$

where  $S_k$  is known in clear from above and other honest  $\llbracket c_{k,i} \rrbracket_2$  can be computed as explained. The adversary  $\mathcal{B}$  then uses  $\llbracket c_{k,H} \rrbracket_2$  to simulation the  $H$ -th key component of the  $k$ -th key query. We emphasize that we makes use of the *static* corruption of  $\mathcal{C}_{\text{sk}_e}$  in the simulation for honest  $i$ , since we never have to compute *all* the  $(c_{k,i})_{i \in \mathcal{H}}$  in the clear and can embed the DDH instance so that on the exponents (of group elements) they sum to  $S_k$ .

It can be verified that if  $c - ab = 0$ , then  $\mathcal{B}$  is simulating the  $k$ -th query where  $\mathcal{B}$  simulates  $\mathbf{k}_{i,\text{ipfe}}[6] = d_{\text{tag-f},y} \theta_i := ab_{k,i}$  and we are in  $\mathbb{G}_{5,k-1}$ ; Else  $\mathbf{d}_{k,i}[6] = d_{\text{tag-f},y} \theta_i := c_{k,i}$  is a totally uniformly random value such that  $\sum_{i \in \mathcal{H}_{\text{sk}_e}} c_{k,i} + d_{\text{tag-f},y} \sum_{i \in \mathcal{C}_{\text{sk}_e}} \theta_i = aS_k + d_{\text{tag-f},y} \sum_{i \in \mathcal{C}_{\text{sk}_e}} \theta_i = 0$  thanks to (8), the fact that we program  $d_{\text{tag-f},y} := a$  for the RO of  $\mathbb{H}_2$ , and the definition of  $S_k$ .

In the end we have  $|\Pr[\mathbb{G}_{5,k-1} = 1] - \Pr[\mathbb{G}_{5,k} = 1]| \leq \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(1^\lambda)$  and thus  $|\Pr[\mathbb{G}_5 = 1] - \Pr[\mathbb{G}_6 = 1]| \leq K \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(1^\lambda)$ .

**Game  $\mathbb{G}_7$ :** We guess the challenge  $\text{tag}$  among the  $Q_1$  queries to the RO and simulate  $\mathbb{H}(\text{tag})$  by a random pair of elements in  $\mathbb{G}_1$ , while for all  $\text{tag}' \neq \text{tag}$  that is queried,  $\mathbb{H}(\text{tag}')$  returns a pair belonging to  $\text{span}((1, \mu)) \subseteq \mathbb{G}_1^2$ . We use the random self-reducibility of DDH, where the running time of the simulator increases by an additive factor  $O(Q_1 \cdot t_{\mathbb{G}_1})$  with  $t_{\mathbb{G}_1}$  being the time for one addition in  $\mathbb{G}_1$  and  $Q_1$  being the number of random oracle queries. We define  $\text{Event}(\text{tag})$  to denote the event where the challenged  $\text{tag}$  is guessed correctly, with probability  $1/Q_1$ . We have

$$|\Pr[\mathbb{G}_7 = 1 \mid \text{Event}(\text{tag})] - \Pr[\mathbb{G}_6 = 1 \mid \text{Event}(\text{tag})]| \leq \text{Adv}_{\mathbb{G}_1}^{\text{DDH}}(1^\lambda) .$$

Notice that  $\Pr[\mathbb{G}_7 = 1 \mid \neg \text{Event}(\text{tag})] = 0$  and the output of  $\mathbb{G}_6$  is independent of  $\text{Event}(\text{tag})$ . Therefore, we have

$$\begin{aligned} \text{Adv}(\mathbb{G}_7) &= \frac{1}{Q_1} \cdot \Pr[\mathbb{G}_7 = 1 \mid \text{Event}(\text{tag})] \\ &\quad + \Pr[\neg \text{Event}(\text{tag})] \Pr[\mathbb{G}_7 = 1 \mid \neg \text{Event}(\text{tag})] - \frac{1}{2} \\ &\geq \frac{1}{Q_1} \cdot \left( \text{Adv}(\mathbb{G}_6) - \text{Adv}_{\mathbb{G}_1}^{\text{DDH}}(1^\lambda) \right) \end{aligned}$$

We thus have  $\text{Adv}(\mathbb{G}_6) \leq \text{Adv}_{\mathbb{G}_1}^{\text{DDH}}(1^\lambda) + Q_1 \cdot \text{Adv}(\mathbb{G}_7)$ .

**Game G<sub>8</sub>**: In this game we virtually change the secret values  $(s_i, u_i)$  to  $(s'_i, u'_i)$  satisfying:

$$\begin{aligned} S' &:= S + \Delta S \\ U' &:= U + \Delta U \end{aligned}$$

where  $(\Delta S, \Delta U)$  satisfies:

$$\begin{cases} (\omega_{\text{tag}}\zeta_1 + \omega'_{\text{tag}}\zeta_2)\Delta S[i] + (\omega'_{\text{tag}}\zeta_3 + \omega_{\text{tag}}\zeta_4)\Delta U[i] = -(\mathbf{x}_b^*[i] - \mathbf{x}_0^*[i]) \\ \Delta S[i](\alpha_i + \gamma_i\frac{\zeta_3}{\zeta_1}\mu) + \Delta U[i](\gamma'_i + \alpha_i\frac{\zeta_3}{\zeta_1}\mu) = 0 \end{cases} \quad (9)$$

This changes the challenge ciphertext into an encryption of  $\mathbf{x}_0^*[i]$  that is independent of  $b$ .

The simulator works as follows:

1. During **Setup**, all secret information are generated for  $\text{msk}$ , including the vectors  $(S, U, V, T)$  and the dual bases  $(H_i, H'_i)_{i \in [n]}$ .
2. For all queries **Corrupt** $(i, \text{skey})$ , thanks to the admissibility condition 6, interpreted particularly for  $\mathcal{F}^{\text{IP}}$  in (7),  $(s'_i, u'_i) = (s_i, u_i)$  for  $i \in \mathcal{C}_{\text{skey}}$  and we can simulate the corrupted  $\text{sk}_i$  in the new form without problems.
3. Because  $\mathcal{C}_{\text{skey}}$  is statically declared up front, the simulator can generate  $(\theta_i)_{i \in \mathcal{C}_{\text{skey}}}$  so as to respond to the corrupted  $\text{sk}_i$ .
4. For all queries to **Enc**, we have

$$\begin{aligned} & -(\chi_{\text{tag}'\zeta_1} + \mu\chi_{\text{tag}'\zeta_2})\mathbf{s}'_i - (\mu\chi_{\text{tag}'\zeta_3} + \chi_{\text{tag}'\zeta_4})\mathbf{u}'_i \\ & = -(\chi_{\text{tag}'\zeta_1} + \mu\chi_{\text{tag}'\zeta_2})s_i - (\mu\chi_{\text{tag}'\zeta_3} + \chi_{\text{tag}'\zeta_4})u_i \end{aligned}$$

by construction of  $(s'_i, u'_i) = (s_i, u_i) + (\Delta S[i], \Delta U[i])$  following system (9).

5. If the adversary queries **Corrupt** $(i, \text{ekey})$  and  $i$  is *not yet* queried to **LoR**: return

$$\begin{aligned} \text{ek}_i &= \\ & (p_i H_i^{(1)} - (\zeta_1 s_i + \zeta_4 u_i) H_i^{(4)}, q_i H_i^{(2)} - (\zeta_2 s_i + \zeta_3 u_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)}). \end{aligned}$$

The response to the query of  $i$  to **LoR** later will be:

$$\begin{aligned} \mathbf{c}_{i, \text{ipfe}} &= \\ & (\omega_{\text{tag}} p_i, \mu \omega_{\text{tag}} q_i, r_i t_i, \\ & -(\omega_{\text{tag}} \zeta_1 + \mu \omega_{\text{tag}} \zeta_2) s'_i - (\mu \omega_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) u'_i + \mathbf{x}_0^*[i] - r_i v_i - \tilde{\rho}_i \tau_i, \tilde{\rho}_i \tau_i, \omega_{\text{tag}}) \mathbf{B}_i. \end{aligned}$$

6. If  $i$  is queried to **LoR** then **Corrupt** $(i, \text{ekey})$  happens afterwards, return:

$$\begin{aligned} \mathbf{c}_{i, \text{ipfe}} &= \\ & (\omega_{\text{tag}} p_i, \mu \omega_{\text{tag}} q_i, r_i t_i, \\ & -(\omega_{\text{tag}} \zeta_1 + \mu \omega_{\text{tag}} \zeta_2) s'_i - (\mu \omega_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) u'_i + \mathbf{x}_0^*[i] - r_i v_i - \tilde{\rho}_i \tau_i, \tilde{\rho}_i \tau_i, \omega_{\text{tag}}) \mathbf{B}_i \\ \text{ek}_i &= \\ & (p_i H_i^{(1)} - (\zeta_1 s'_i + \zeta_4 u'_i) H_i^{(4)}, q_i H_i^{(2)} - (\zeta_2 s'_i + \zeta_3 u'_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)}) \end{aligned}$$

where  $s'_i := s_i + \Delta S[i]$ ,  $u'_i := u_i + \Delta U[i]$  defined by (9).

7. For all  $i \in \mathcal{H}_{\text{ekey}}$  whose  $\text{ek}_i$  is never revealed, in the end the simulator sets

$$\begin{aligned} \text{ek}_i &= \\ & (p_i H_i^{(1)} - (\zeta_1 s'_i + \zeta_4 u'_i) H_i^{(4)}, q_i H_i^{(2)} - (\zeta_2 s'_i + \zeta_3 u'_i) H_i^{(4)}, t_i \mathbf{h}_{i,3} - v_i \mathbf{h}_{i,4}, \mathbf{h}_{i,4}, H_i^{(6)}). \end{aligned}$$

The same goes for  $i \in \mathcal{H}_{\text{skey}}$ .

8. For all functional key queries to **DKeyGenShare**, if  $i \in \mathcal{H}_{\text{skey}}$ , the vector  $\mathbf{k}_{i, \text{ipfe}}$  is responded using  $(s_i, u_i)$ :

$$\mathbf{k}_{i, \text{ipfe}} = ((s_i \alpha_i + u_i \gamma'_i) y_i, (s_i \gamma_i + u_i \alpha_i) y_i, \frac{v_i}{t_i} y_i, y_i^{(\ell)}, y_i, d_{\text{tag-f,y,i}}) \mathbf{H}_i^*$$

following the changes from G<sub>6</sub>.

Our goal now is to argue that under the SXDH assumption, the above simulator can simulate  $\mathbf{G}_8$  in a computationally indistinguishable manner compared to  $\mathbf{G}_7$ . The main challenge is to ensure that the correctness of decryption is retained, taking into account the fact that the adversary can use  $\mathbf{ek}_i$  to craft their own  $\mathbf{c}_{i,\text{ipfe}}$  as well as  $\mathbf{sk}_i$  to craft their own  $\mathbf{k}_{i,\text{ipfe}}^{(\ell)}$ , while the corrupted clients  $\mathcal{C}_{\text{sk}}^{\text{key}}$  are declared statically at the beginning of the game. From the system (9), for  $i \in \mathcal{C}_{\text{ekey}}$ , if  $\Delta \mathbf{x}[i] = \mathbf{x}_1^*[i] - \mathbf{x}_0^*[i] \neq 0$ , then it could be the case that  $(s'_i, u'_i) \neq (s_i, u_i)$  and

$$\begin{aligned} \mathbf{k}_{i,\text{ipfe}}^{(\ell)} &= ((s_i \alpha_i + u_i \gamma'_i) y_i^{(\ell)}, (s_i \gamma_i + u_i \alpha_i) y_i^{(\ell)})_{\mathbf{H}_i^*[1,2]} \\ &= ((s'_i \alpha_i + u'_i \gamma'_i) y_i^{(\ell)} - (\Delta S[i] \alpha_i + \Delta U[i] \gamma'_i) y_i^{(\ell)}, \\ &\quad (s'_i \gamma_i + u'_i \alpha_i) y_i^{(\ell)} - (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) y_i^{(\ell)})_{\mathbf{H}_i^*[1,2]} \end{aligned}$$

does not have the correct form w.r.t  $(S', U')$ . Our idea is to do a computational basis change to get rid of the errors introduced by  $(\Delta S[i], \Delta U[i])$  using the SXDH assumption. Noting that for all  $i \in \mathcal{C}_{\text{sk}}^{\text{key}}$ ,  $\Delta \mathbf{x}[i] = \mathbf{x}_1^*[i] - \mathbf{x}_0^*[i] = 0$  and we thus have to perform the basis changes only for  $i \in \mathcal{H}_{\text{ekey}} = [n] \setminus \mathcal{C}_{\text{sk}}^{\text{key}}$  at **Setup** time, given the statically declared  $\mathcal{C}_{\text{sk}}^{\text{key}}$ . We first describe how the basis are changed and then detail how the correctness is preserved. We denote by  $(p_1, \dots, p_n)$  the values that are generated at **Setup** and satisfy (5).

For each  $i \in \mathcal{H}_{\text{sk}}^{\text{key}}$ , given a DSDH instance  $(\llbracket a \rrbracket_2, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2)$  where  $\rho := c - ab$  is either 0 or 1, the simulator performs a basis change using the matrices  $(H_i, H'_i)$  defined as below:

$$\begin{aligned} H_i &:= \begin{bmatrix} 1 & 0 & \frac{\mu a \omega_{\text{tag}}}{p_i} \\ 0 & 1 - a \frac{\zeta_1 \omega_{\text{tag}}}{\zeta_3 p_i} \\ 0 & 0 & 1 \end{bmatrix}_{1,2,6} & H'_i &:= (H_i^{-1})^\top = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{-\mu a \omega_{\text{tag}}}{p_i} & a \frac{\zeta_1 \omega_{\text{tag}}}{\zeta_3 p_i} & 1 \end{bmatrix}_{1,2,6} \\ \mathbf{H}_i &= H_i \cdot \mathbf{T}; & \mathbf{H}_i^* &= H'_i \cdot \mathbf{T}^* . \end{aligned}$$

This will change  $(\mathbf{h}_{i,1}, \mathbf{h}_{i,2})$  and  $\mathbf{h}_{i,6}^*$ . However, as  $\mathbf{h}_{i,6}^*$  is the hidden basis vector for  $i \in \mathcal{H}_{\text{sk}}^{\text{key}}$  and  $H_i^{(1)}, H_i^{(2)}$  appear in  $\mathbf{ek}_i$  only as part of random linear combinations, the changes are not recognized by the adversary. We do not have  $\llbracket a \rrbracket_1$  to compute individual vectors  $\mathbf{h}_{i,1}, \mathbf{h}_{i,2}$  but the simulation of the encryption oracles concerns solely the combination  $p_i \mathbf{h}_{i,1} + q_i \mu \mathbf{h}_{i,2}$  and by noting that  $\zeta_1 / \zeta_3 = p_i / q_i$ , we have  $p_i \mathbf{h}_{i,1} + q_i \mu \mathbf{h}_{i,2}$  stays invariant under this basis change. Therefore the simulation can still be performed.

The ciphertexts component from **LoR** can be written in  $\mathbf{T}$ , because we do not have individual vectors  $\mathbf{h}_{i,1}, \mathbf{h}_{i,2}$ , to see how they will be affected:

$$\begin{aligned} \mathbf{c}_{i,\text{ipfe}} &= (p_i \omega_{\text{tag}}, q_i \omega'_{\text{tag}}, \omega_{\text{tag}})_{\mathbf{T}[1,2,6]} \\ &= \left( p_i \omega_{\text{tag}}, q_i \omega'_{\text{tag}}, \omega_{\text{tag}} - a \mu \omega_{\text{tag}} \omega_{\text{tag}} + a \frac{\zeta_1 \omega_{\text{tag}}}{\zeta_3 p_i} q_i \omega'_{\text{tag}} \right)_{\mathbf{H}_i[1,2,6]} \\ &= (p_i \omega_{\text{tag}}, q_i \omega'_{\text{tag}}, \omega_{\text{tag}} + a \omega_{\text{tag}} (\omega'_{\text{tag}} - \mu \omega_{\text{tag}}))_{\mathbf{H}_i[1,2,6]} \end{aligned}$$

where  $p_i, q_i \stackrel{\$}{\leftarrow} \mathbb{Z}_q$  are chosen by the simulator at **Setup** satisfying (5). On the other hand, for the ciphertexts component from **Enc**, we can write them in  $\mathbf{T}$  and the same calculation demonstrates that they retains their normal form required for the **Enc** thanks to the programming of the RO.

We now consider the correction of the partial functional key component  $\mathbf{k}_{i,\text{ipfe}}$ , where  $i \in \mathcal{H}_{\text{sk}}^{\text{key}}$ . We emphasize again that we are using the fact  $\mathcal{C}_{\text{sk}}^{\text{key}}$  is statically declared, all basis changes (defined during **Setup**) are done for  $i \in \mathcal{H}_{\text{sk}}^{\text{key}}$ , and this implies we know at the time of answering **DKeyGenShare** which components are honest:

$$\begin{aligned}
& \mathbf{k}_{i,\text{ipfe}} \\
&= (s'_i y_i \alpha_i + u'_i y_i \gamma'_i - \Delta S[i] y_i \alpha_i - \Delta U[i] y_i \gamma'_i, \\
&\quad s'_i y_i \gamma_i + u'_i y_i \alpha_i - \Delta S[i] y_i \gamma_i - \Delta U[i] y_i \alpha_i, \\
&\quad d_{\text{tag-f,y},i})_{\mathbf{H}_i^*[1,2,6]} \\
&+ \frac{\zeta_3}{\zeta_1} (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) \cdot (-\mu c y_i, \frac{\zeta_1}{\zeta_3} c y_i, 0, b p_i y_i / \theta_i)_{\mathbf{T}^*[1,2,6]} \\
&= (s'_i y_i \alpha_i + u'_i y_i \gamma'_i - \Delta S[i] y_i \alpha_i - \Delta U[i] y_i \gamma'_i, \\
&\quad s'_i y_i \gamma_i + u'_i y_i \alpha_i - \Delta S[i] y_i \gamma_i - \Delta U[i] y_i \alpha_i, \\
&\quad d_{\text{tag-f,y},i})_{\mathbf{H}_i^*[1,2,6]} \\
&+ \frac{\zeta_3}{\zeta_1} (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) \cdot (-\mu \rho y_i, \frac{\zeta_1}{\zeta_3} \rho y_i, 0, b p_i y_i / \omega_{\text{tag}})_{\mathbf{H}_i^*[1,2,6]} \\
&= (s'_i y_i \alpha_i + u'_i y_i \gamma'_i - \Delta S[i] y_i \alpha_i - \Delta U[i] y_i \gamma'_i - \frac{\zeta_3}{\zeta_1} (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) \cdot \mu \rho y_i, \\
&\quad s'_i y_i \gamma_i + u'_i y_i \alpha_i - \Delta S[i] y_i \gamma_i - \Delta U[i] y_i \alpha_i + (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) \cdot \rho y_i, \\
&\quad d_{\text{tag-f,y},i} + \frac{\zeta_3}{\zeta_1} (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) \cdot b p_i y_i / \omega_{\text{tag}})_{\mathbf{H}_i^*[1,2,6]}
\end{aligned}$$

If  $\rho = 0$  we are not correcting the vector, else if  $\rho = 1$ , the first two coordinates of  $\mathbf{k}_{i,\text{ipfe}}^{(\ell)}$  are  $(s'_i y_i^{(\ell)} \alpha_i + u'_i y_i^{(\ell)} \gamma'_i, s'_i y_i^{(\ell)} \gamma_i + u'_i y_i^{(\ell)} \alpha_i)$  as desired. We recall that the 6-th coordinate of  $\mathbf{k}_{i,\text{ipfe}}$  contains independent random secret sharings (together with the corrupted  $\text{sk}_i$ ) thanks to  $\mathbf{G}_6$ .

It is important to note that we are able to simulate  $\mathbf{k}_{i,\text{ipfe}}^{(\ell)}$  only for  $i \in [n] \setminus \mathcal{C}_{\text{sk}}^{\text{key}}$  because for  $i \in \mathcal{C}_{\text{sk}}^{\text{key}}$  the adversary can use  $\text{sk}_i$  to craft their own  $\mathbf{k}_{i,\text{ipfe}}^{(\ell)}$ . This is feasible for a correct simulation thanks to the fact that  $\mathcal{C}_{\text{sk}}^{\text{key}}$  is static and the constraint that  $\Delta \mathbf{x}[i] = 0$  for all  $i \in \mathcal{C}_{\text{sk}}^{\text{key}}$ . Regarding the challenge component  $\mathbf{c}_{i,\text{ipfe}}$ , there will always be the additional term  $a \omega_{\text{tag}} (\omega'_{\text{tag}} - \mu \omega_{\text{tag}})$  because all computation is done in  $\mathbf{T}$  then the change carries the vector to  $\mathbf{H}_i$ , no matter whether  $\mathbf{c}_{i,\text{ipfe}}$  comes from  $\mathbf{LoR}$  or is computed by a corrupted  $\text{ek}_i$ . We are implicitly using the fact that all key components  $\mathbf{k}_{\text{tag-f,y}}$  must be consistent w.r.t  $d_{\text{tag-f,y}}$  so as to regroup and obtain the 0-sum  $\sum_{i=1}^n d_{\text{tag-f,y},i}$  when decrypting.

We now verify that the decryption's correctness is preserved:

$$\begin{aligned}
& (\mathbf{ct}_{\text{tag},i} \times \mathbf{k}_{i,\text{ipfe}}) \\
&= \left( \begin{array}{c} (\omega p_i, \omega' q_i, r_i t_i, -(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i - r_i v_i - \tilde{\rho}_i \tau_i, \\ \tilde{\rho}_i \tau_i, \omega_{\text{tag}} (1 + a(\omega'_{\text{tag}} - \mu \omega_{\text{tag}})))_{\mathbf{H}_i} \\ \times \\ (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, -\frac{v_i}{t_i} y_i, y_i, \\ y_i, d_{\text{tag-f,y},i} + \frac{\zeta_3}{\zeta_1} (\Delta S[i] \gamma_i + \Delta U[i] \alpha_i) \cdot b p_i y_i^{(\ell)} / \omega_{\text{tag}})_{\mathbf{H}_i^*} \end{array} \right)
\end{aligned}$$

and thus there will exist an additional term (from the 6-th component)

$$\left[ (y_i \Delta S[i] \zeta_2 + y_i \Delta U[i] \zeta_3) \cdot ab \cdot (\omega'_{\text{tag}} - \mu \omega_{\text{tag}}) \right]_{\mathbf{T}} \quad (10)$$

while we are using the system (5) to get rid of the scalars  $(p_i, q_i, \alpha_i, \gamma_i)$ . Due to the specification of  $(\Delta S[i], \Delta U[i])$  from system (9), we can deduce that  $(\Delta S[i], \Delta U[i])$  is a multiple of  $\Delta \mathbf{x} = \mathbf{x}_b^* - \mathbf{x}_0^*$ . Then, summing all  $\mathbf{ct}_{\text{tag},i} \times \mathbf{k}_{i,\text{ipfe}}$  over  $i \in [n]$  will give  $\langle \Delta S, \mathbf{y} \rangle$  and  $\langle \Delta U, \mathbf{y} \rangle$ , by noting that for  $i \in \mathcal{C}_{\text{sk}}^{\text{key}}$  we have  $\Delta S[i] = \Delta U[i] = 0$  because  $\Delta \mathbf{x}[i] = 0$ . Therefore, at the end of decryption, the noisy term of (10) will give us a multiple of  $\langle \Delta \mathbf{x}, \mathbf{y} \rangle$ , which evaluates to 0 due to the adversary's admissibility. This implies the correctness is preserved.

Last but not least, if  $i \in [n] \setminus \mathcal{C}_{\text{sk}}^{\text{key}}$  but  $y_i^{(\ell)} = 0$  and  $\Delta \mathbf{x}[i] \neq 0$  then we will have a trivial DSDH. We note that, for instance, this case might happen for a corrupted  $i \in \mathcal{C}_{\text{ek}} \setminus \mathcal{C}_{\text{sk}}^{\text{key}}$  of which  $\mathbf{c}_{i,\text{ipfe}}$  must be queried to  $\mathbf{LoR}$  in the challenge query (and the adversary receives  $n$  corresponding components). We recall that the corruption of  $i$  can be before or after the  $\mathbf{LoR}$  query. Nevertheless, this does not compromise the simulation and the indistinguishability, for the key component of  $i$  is now:

$$\mathbf{k}_{i,\text{ipfe}} = (0, 0, 0, 0, d_{\text{tag-f,y},i})_{\mathbf{H}_i^*}$$

and the corresponding challenge ciphertext component  $\mathbf{c}_{i,\text{ipfe}}$  from  $\mathbf{LoR}$  is



$$\mathbf{c}_{i,\text{ipfe}} = (\dots, r_i t_i, -(\omega_{\text{tag}} \zeta_1 + \omega'_{\text{tag}} \zeta_2) s'_i - (\omega'_{\text{tag}} \zeta_3 + \omega_{\text{tag}} \zeta_4) u'_i + \mathbf{x}_0^*[i] - r_i v_i - \tilde{\rho}_i \tau_i, \tilde{\rho}_i \tau_i, \omega_{\text{tag}}) \mathbf{H}_i.$$

The correctness of decryption is not affected, while thanks to the perfect masking by  $\tilde{\rho}_i \tau_i$  that is uniformly random and independent of  $(\mathbf{x}_0^*[i], \mathbf{x}_1^*[i])$ ,  $b$  is perfectly hidden by  $\mathbf{c}_{i,\text{ipfe}}$ . This means that  $\mathbf{c}_{i,\text{ipfe}}$  is perfect indistinguishable from the previous game. We need at most  $n$  instances of DSDH for the basis change and in the end  $|\text{Adv}(\mathbf{G}_8) - \text{Adv}(\mathbf{G}_7)| \leq 2 \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(1^\lambda)$ .

In  $\mathbf{G}_7$  the challenge bit  $b$  is perfectly hidden: all challenge ciphertext components returned from **LoR** do not depend on  $b$  and if for  $i \in \mathcal{C}_{\text{ekey}} \setminus \mathcal{C}_{\text{skey}}$ , where  $\Delta \mathbf{x}[i] \neq 0$  and  $y_i = 0$ , the resulting  $\mathbf{c}_{i,\text{ipfe}}$  perfectly hides  $b$ . Therefore  $\text{Adv}(\mathbf{G}_8) = 0$  and in total

$$\begin{aligned} \text{Adv}_{\mathcal{E}, \mathcal{F}, \mathcal{A}}^{\text{dmc-stat-ind-cpa}}(1^\lambda) &\leq \sum_{i=0}^7 |\text{Adv}(\mathbf{G}_{i+1}) - \text{Adv}(\mathbf{G}_i)| \\ &\leq 3 \cdot \text{Adv}_{\mathbb{G}_1}^{\text{DDH}}(1^\lambda) + (2Q_1 + K) \cdot \text{Adv}_{\mathbb{G}_2}^{\text{DDH}}(1^\lambda) + \frac{Q_2^2}{2q} \end{aligned}$$

and the proof is completed.  $\square$

### C.3 Security of Our DMCFE Construction in Section 6.1

**Theorem 26.** *Let  $\mathcal{E} = (\text{Setup}, \text{DKShare}, \text{DKeyComb}, \text{Enc}, \text{Dec})$  be the DMCFE constructed in Section 6.1. Then,  $\mathcal{E}$  is one-time statically IND+-secure in the ROM following the security model in Definition 18 if the SXDH and DBDH assumptions hold for  $\mathbb{G}_1$  and  $\mathbb{G}_2$ . More specifically, let  $n$  denote the dimension for inner-products,  $Q_1, Q_2$  denote the maximum number of random oracle (RO) queries to  $\mathbf{H}_1, \mathbf{H}_2$  and  $K$  denote the total number of functional key queries. For any one-time challenge ppt adversary  $\mathcal{A}$  against  $\mathcal{E}$  with static corruption of secret keys and encryption keys, we have the following bound:*

$$\text{Adv}_{\mathcal{E}, \mathcal{F}^{\text{IP}}, \mathcal{A}}^{\text{dmc-stat-1chal+}}(1^\lambda) \leq (K + 1) \text{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\text{DBDH}}(1^\lambda) + (3 + 2Q_1 + K) \text{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\text{SXDH}}(1^\lambda) + \frac{Q_2^2}{2q}.$$

*Proof (Of Theorem 26).* The advantage of an adversary  $\mathcal{A}$  in a game  $\mathbf{G}_i$  is denoted by

$$\text{Adv}(\mathbf{G}_i) := |\Pr[\mathbf{G}_i = 1] - 1/2|$$

where the probability is taken over the random choices of  $\mathcal{A}$  and coins of  $\mathbf{G}_i$ .

**Game  $\mathbf{G}_0$ :** This is the security game  $\text{Expr}_{\mathcal{E}^{(+)}, \mathcal{F}^{\text{IP}}, \mathcal{A}}^{\text{dmc-stat-1chal+}}(1^\lambda)$ . The adversary will declare the sets  $\mathcal{C}_{\text{skey}}, \mathcal{C}_{\text{ekey}}$  of corrupted secret keys  $\text{sk}_i$  and corrupted encryption keys  $\text{ek}_i$  at the beginning of the game. We have  $\text{Adv}_{\mathcal{E}^{(+)}, \mathcal{F}^{\text{IP}}, \mathcal{A}}^{\text{dmc-stat-1chal+}}(1^\lambda) = \text{Adv}(\mathbf{G}_0)$ .

**Game  $\mathbf{G}_1$ :** We change how the responses from **DKeyGenShare** are generated. Let  $\ell \in \{1, \dots, K\}$  index the  $\ell$ -th functional key, which is composed of  $(\mathbf{k}_{i,\text{ipfe}}^{(\ell)})_{i \in [n]}$ . The goal of this game is to keep all *complete* functional keys normal, while switching the *incomplete* functional keys to simulated keys of random vectors.

More specifically, if the  $\ell$ -th key query for  $(\text{tag-f}, \mathbf{y}^{(\ell)})$  is incomplete, then all components  $(\mathbf{k}_{j,\text{ipfe}}^{(\ell)})_j$  are generated by  $\text{DKShare}(\text{sk}_i, \text{tag-f}, \mathbf{r})$  for some uniformly random vector  $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_q^n$ . We proceed by a sequence of hybrids  $\mathbf{G}_{0,\ell}$  for  $\ell \in [K]$ , where in  $\mathbf{G}_{0,\ell}$  all the  $\ell'$ -th keys for  $\ell' \leq \ell$  is satisfying the above simulation, and all the  $\ell'$ -th keys for  $\ell' > \ell$  are still correctly generated as in  $\mathbf{G}_0$ . It holds that  $\mathbf{G}_{0,K} = \mathbf{G}_1$ .

**From  $\mathbf{G}_{0,\ell-1}$  to  $\mathbf{G}_{0,\ell}$ :** We use the DBDH assumption to argue the difference in advantages  $|\text{Adv}(\mathbf{G}_{0,\ell}) - \text{Adv}(\mathbf{G}_{0,\ell-1})|$ . Given  $([1], [a]_1, [b]_1, [b]_2, [c]_2, [d]_t)$  as a DBDH instance where  $d - abc = 0$  or a uniformly random value, we want to construct a simulator that simulates the game  $\mathbf{G}_{0,\ell-1}$  for the adversary if  $d - abc = 0$  and  $\mathbf{G}_{0,\ell}$  if  $d - abc$  is a uniformly random value. More specifically, our goal is to construct a simulator of the IND-security game using the DBDH instance such that:

1. If the  $\ell$ -th key is complete, then it stays correctly answered no matter  $d - abc = 0$  or a uniformly random value.
2. If the  $\ell$ -th key is incomplete, then it is a correct key for  $(\text{tag-f}, \mathbf{y}^{(\ell)})$  when  $d - abc = 0$  and a key of some random function when  $d - abc$  is a uniformly random value.

During **Setup**, all secret information is generated and the simulator also generates a secret sharing  $(\tilde{\epsilon}_i)_i$  of 0 over  $i \in \mathcal{H}_{\text{skey}}$ . Concerning the values  $(\epsilon_i)_i$ , the simulator first samples  $\epsilon_i \xleftarrow{\$} \mathbb{Z}_q$  for  $i \in \mathcal{C}_{\text{skey}}$  which is given statically. Because  $\mathcal{C}_{\text{skey}}$  is statically declared, the simulator is using basis changes for  $(\mathbf{H}_i, \mathbf{H}_i^*)$  with  $i \in \mathcal{H}_{\text{skey}}$  w.r.t the matrices:

$$\begin{aligned} H_i &:= \begin{bmatrix} 1 & 0 \\ -\tilde{\epsilon}_i c / \theta_i & 1 \end{bmatrix}_{6,8} & H'_i &:= (H_i^{-1})^\top = \begin{bmatrix} 1 & \tilde{\epsilon}_i c / \theta_i \\ 0 & 1 \end{bmatrix}_{6,8} \\ \mathbf{H}_i &= H_i \cdot \mathbf{T}; & \mathbf{H}_i^* &= H'_i \cdot \mathbf{T}^* . \end{aligned}$$

The RO for  $\mathbf{H}_2$  is also programmed so that  $\mathbf{H}_2(\text{tag-f}, \mathbf{y}^{(\ell)}) \rightarrow \llbracket b \rrbracket_2$ . Then, for all **DKeyGenShare** query  $(i, \text{tag-f}, \mathbf{y}^{(\ell)}[i])$ , the simulator replies with  $\text{dk}_{\text{tag-f}, \mathbf{y}, i} := (\mathbf{k}_{i, \text{ipfe}}, \epsilon \cdot \mathbf{t}_8, \llbracket \epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}} + \tilde{\epsilon}_i d \rrbracket_{\mathbf{t}})$  where

$$\begin{aligned} &\mathbf{k}_{i, \text{ipfe}} \\ &= (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, -\frac{v_i}{t_i} y_i, y_i, 0, 0, 0, 0, -e_i \kappa_{\text{tag-f}, \mathbf{y}}) \mathbf{H}_i^* \\ &+ (0, 0, 0, 0, 0, b \theta_i, 0, 0) \mathbf{T}^* \\ &= (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, -\frac{v_i}{t_i} y_i, y_i, 0, 0, 0, 0, -e_i \kappa_{\text{tag-f}, \mathbf{y}}) \mathbf{H}_i^* \\ &+ (0, 0, 0, 0, 0, b \theta_i, 0, -\tilde{\epsilon}_i b c) \mathbf{H}_i^* \\ &= (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, -\frac{v_i}{t_i} y_i, y_i, 0, \kappa_{\text{tag-f}, \mathbf{y}} \theta_i, 0, -e_i \kappa_{\text{tag-f}, \mathbf{y}} - \tilde{\epsilon}_i b c) \mathbf{H}_i^* \\ \mathbf{e}(\epsilon_i \cdot \llbracket \langle E, \mathbf{1} \rangle \rrbracket_1, \llbracket \kappa_{\text{tag-f}, \mathbf{y}} \rrbracket_2) &= \llbracket \epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}} \rrbracket_{\mathbf{t}} . \end{aligned}$$

The value  $a \cdot \mathbf{t}_8$  is computed using  $\llbracket a \rrbracket_1$  and  $T^{(8)}$ . If the current index  $i^*$  is the *last* index in  $\mathcal{H}_{\text{skey}}$ , i.e. the ongoing  $\ell$ -th key is complete, we simulate  $\epsilon_{i^*} := a - \sum_{j \neq i^*} \epsilon_j$  and  $\llbracket \epsilon_{i^*} \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}} \rrbracket_{\mathbf{t}}$  can be computed using  $\llbracket a \rrbracket_1, \llbracket b \rrbracket_2$ .

We write it in the basis  $\mathbf{T}$ , which is a random basis and indistinguishable from writing using a vector of  $\mathbf{H}_i$ . We will verify the two conditions **1** and **2**:

- Suppose the  $\ell$ -th key is *complete*, i.e. all  $i \in \mathcal{H}_{\text{skey}}$  are queried. We first notice that:

$$\llbracket a \rrbracket_1 \cdot T^{(8)} = (0^5, \tilde{\epsilon}_i a c / \theta_i, 0, a) \mathbf{H}_i$$

by the basis change. Therefore, the decryption procedure computes

$$\begin{aligned} &\sum_{i=1}^n \left( \begin{array}{c} (s_i y_i \alpha_i + u_i y_i \gamma'_i, s_i y_i \gamma_i + u_i y_i \alpha_i, -\frac{v_i}{t_i} y_i, y_i, \\ 0, \kappa_{\text{tag-f}, \mathbf{y}} \theta_i, \delta, -e_i \kappa_{\text{tag-f}, \mathbf{y}} - \tilde{\epsilon}_i b c) \mathbf{H}_i^* \\ \times \\ (\omega p_i, \omega' q_i, r_i t_i, -(\omega \zeta_1 + \omega' \zeta_2) \cdot s_i - (\omega' \zeta_3 + \omega \zeta_4) \cdot u_i + x_i - r_i v_i, \\ 0, \omega + \tilde{\epsilon}_i a c / \theta_i, -d_i \omega, a) \mathbf{H}_i \end{array} \right) \\ &+ \sum_{i=1}^n \llbracket \epsilon_i \langle E, \mathbf{1} \rangle \kappa_{\text{tag-f}, \mathbf{y}} + \tilde{\epsilon}_i d \rrbracket_{\mathbf{t}} + \llbracket \delta \langle D, \mathbf{1} \rangle \omega \rrbracket_{\mathbf{t}} . \end{aligned}$$

Thanks to the definition of the response to the *last* index  $i^* \in \mathcal{H}_{\text{skey}}$  that is queried to **DKeyGenShare**, as well as the fact that  $(\theta_i)_i, (\tilde{\epsilon}_i)_i$  are secret sharing of 0, we can verify that the correctness of decryption persists.

- Suppose the  $\ell$ -th key is *incomplete*, the same computation, with indices in a strict subset  $\tilde{\mathcal{I}} \subset \mathcal{H}_{\text{skey}}$  will lead to

$$(d - abc) \cdot \left( \sum_{i \in \tilde{\mathcal{I}}} \tilde{\epsilon}_i \right) .$$

Moreover, we never reach the last index  $i^*$  so as to simulate a key component that is different from the correct result of **DKeyGenShare**. If  $d = ab \cdot c$  we are simulating the key components of the  $\ell$ -th key forming correct components for  $(\text{tag-f}, \mathbf{y}^{(\ell)})$  and we are in

$G_{0,\ell-1}$ . Otherwise,  $d$  is a uniformly random value and the  $\ell$ -th key is identically distributed as a key for some random vector and we are in  $G_{0,\ell}$ . We also remark that in the latter case where  $d \xleftarrow{\$} \mathbb{Z}_q$ , the term  $(d - abc) \cdot (\sum_{i \in \bar{J}} \tilde{\epsilon}_i)$  acts as a one-time pad mask during decryption and perfectly hide the challenge bit conditioned on using this incomplete  $\ell$ -th key.

In the end, we have  $|\text{Adv}(G_{0,\ell}) - \text{Adv}(G_{0,\ell-1})| \leq \text{Adv}_{G_1, G_2}^{\text{DBDH}}(1^\lambda)$  and thus  $|\text{Adv}(G_1) - \text{Adv}(G_0)| = |\text{Adv}(G_{0,K}) - \text{Adv}(G_0)| \leq K \times \text{Adv}_{G_1, G_2}^{\text{DBDH}}(1^\lambda)$ .

**Game  $G_2$ :** We change how the responses from **LoR** are generated. If the challenge queries are incomplete, i.e. the set  $J$  of indices in  $\text{LoR}((x_j^{(0)}, x_j^{(1)})_{j \in J \subseteq [n]}, \text{tag}^*)$  does not contain  $\mathcal{H}_{\text{ekey}}$ , using a similar reduction to DBDH as in the case of incomplete key queries, by accumulating a secret sharing of 0 into the *last* index  $i^* \in \mathcal{H}_{\text{ekey}}$  that will *not* be queried. This means we have a one-time pad mask and the incomplete challenge ciphertext can be viewed as encrypting a constant independent of  $b$ . Otherwise, the challenge ciphertext components are responded normally. We recall that because  $\mathcal{H}_{\text{ekey}}$  is statically declared at the beginning of the game. We again rely on the DBDH assumption to argue the difference in advantages. The simulation is identical, except that we do not need multiple hybrids because we are in the *one-time IND-security* game. In the end, we have  $|\text{Adv}(G_2) - \text{Adv}(G_1)| \leq \text{Adv}_{G_1, G_2}^{\text{DBDH}}(1^\lambda)$ .

**Game  $G_3$ :** We now apply the security of the underlying DMCFE scheme to modify the functional keys that are complete and at the end switch the challenge ciphertext to encryption of  $\mathbf{x}_0^*$  if it is complete. All the basis changes are performed as in Theorem 16, where we only pay attention to complete functional keys during simulation. Following the above games, if the challenge queries are incomplete, the responses are perfectly masked and independent of  $b$ . If they are complete, we are able to switch them to encryptions of  $\mathbf{x}_0^*$  with negligible differences in advantages. Due to Theorem 16, it holds that

$$|\text{Adv}(G_3) - \text{Adv}(G_2)| \leq (3 + 2Q_1 + K) \cdot \text{Adv}_{G_1, G_2}^{\text{SXDH}}(1^\lambda) + \frac{Q_2^2}{2q} .$$

In  $G_3$ , the challenge ciphertexts fully hides  $b$ , either they are complete or not. Therefore  $\text{Adv}(G_3) = 0$  and the proof is completed.  $\square$