Function-Hiding Decentralized Multi-Client Functional Encryption for Inner Products

Ky Nguyen, David Pointcheval, and Robert Schädlich

DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France

Abstract. Decentralized Multi-Client Functional Encryption (DMCFE) extends the basic functional encryption to multiple clients that do not trust each other. They can independently encrypt the multiple inputs to be given for evaluation to the function embedded in the functional decryption key. And they keep control on these functions as they all have to contribute to the generation of the functional decryption keys.
As any encryption scheme, all the FE schemes provide privacy of the plaintexts. But the functions associated to the functional decryption keys might be sensitive too (*e.g.* a model in machine learning). The function-hiding property has thus been introduced to additionally protect the function evaluated during the decryption process. But it was not properly defined for previous definitions of DMCFE.
In this paper, we provide a formal definition of DMCFE with complete function-hiding security game. We thereafter propose a concrete construction of function-hiding DMCFE for inner products, with strong security guarantees: the adversary is allowed to adaptively query multiple challenge ciphertexts and multiple challenge keys. Previous constructions were proven secure for a single challenge ciphertext only, in the selective setting (*i.e.* provided before the setup).

Keywords: Functional Encryption, Inner Product, Function-Hiding

1 Introduction

Functional Encryption. Public-Key Encryption (PKE) has become so indispensable that without this building block, secure communication over the Internet would be unfeasible nowadays. However, this concept of PKE limits the access to encrypted data in an *all-or-nothing* fashion: once the recipients have the secret key, they will be able to recover the original data; otherwise, no information is revealed. The concept of Functional Encryption (FE), originally introduced by Boneh, Sahai and Waters [48, 20], overcomes this limitation: a decryption key can be generated under some specific function F, namely a *functional decryption key*, and enable the evaluation $F(x)$ from an encryption of a plaintext x in order to provide a finer control over the leakage of information about x.
Since its introduction, FE has provided a unified framework for prior advanced encryption notions, such as Identity-Based Encryption [49, 27, 19] or Attribute-Based Encryption [48, 36, 47, 14, 46], and has become a very active domain of research. Abdalla *et al.* [3] proposed the first FE scheme (ABDP scheme) that allows computing the inner product between a functional vector in the functional decryption key and a data vector in the ciphertext, coined IPFE. The interests in FE then increased, either in improving existing constructions for concrete function classes, *e.g.* inner products [11, 16, 21] and quadratic functions [15, 32, 13, 41], or in pushing the studies of new advanced notions [34] as well as the relationship to other notions in cryptography [12, 18]. While FE with a single encryptor, *i.e.* single-client FE, is of great theoretical interest, there is also a motivation to investigate a multi-user setting, which might be applicable in practical applications when the data is an aggregation of information coming from multiple sources.

Extensions of FE in the Multi-User Setting. Goldwasser *et al.* [33, 35] initiated the study of *Multi-Input Functional Encryption* (MIFE) and *Multi-Client Functional Encryption* (MCFE). In MCFE particularly, the encrypted data is broken into a vector $(x_1, \ldots, x_n)$ and a client i among n

clients uses their encryption key ek_i to encrypt x_i, under some (usually time-based) tag tag. Given a vector of ciphertexts $(\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{ek}_1, \mathsf{tag}, x_1), \ldots, \mathsf{ct}_n \leftarrow \mathsf{Enc}(\mathsf{ek}_n, \mathsf{tag}, x_n))$, a decryptor holding a functional decryption key dk_F can decrypt and obtain $F(x_1, \ldots, x_n)$ as long as all $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ are generated under the same tag. No information beyond $F(x_1, \ldots, x_n)$ is leaked, especially concerning the individual secret component x_i, and combinations of ciphertexts under different tags provide no further information either. Furthermore, encrypting x_i under different $\mathsf{tag}' \neq \mathsf{tag}$ might bear a different meaning with respect to a client i and thus controls the possibilities constituting ciphertext vectors[1]. This necessitates the encryption keys ek_i being private. The notion of MCFE can be seen as an extension of FE where multiple clients can contribute into the ciphertext vector independently and non-interactively, where encryption is done by private encryption keys. After their introduction, MIFE/MCFE motivated a plethora of works on the subject, notably for the concrete function class of inner products [30, 24, 25, 4, 2, 1, 40, 26, 5, 43].

Decentralized Multi-Client Functional Encryption The setup of MCFE requires some authority (a trusted third party) responsible for the setup and generation of functional decryption keys. The authority possesses a master secret key msk that can be used to handle the distribution of private encryption keys ek_i and deriving functional decryption keys dk_F. When clients do not trust each other, this centralized setting of authority might be a disadvantage. The need for such a central authority is completely eliminated in the so-called *Decentralized Multi-Client Functional Encryption* (DMCFE) introduced by Chotard *et al.* [24]. In DMCFE, only during the setup phase do we need interaction for generating parameters that will be needed by the clients later. The key generation is done independently by different senders, each has a *secret key* sk_i. Agreeing on a function F, each sender generates their partial functional key $\mathsf{dk}_{F,i}$ using sk_i, the description of F, and a tag $\mathsf{tag\text{-}f}$. Originally in [24], the tag $\mathsf{tag\text{-}f}$ can contain the description of F itself. Using DMCFE, the need of an authority for distributing functional keys is completely removed, with minimal interaction required during setup. The seminal work of [24] constructed the first DMCFE for computing inner products, where n clients can independently contribute to the ciphertext vector $(\mathsf{ct}_1 \leftarrow \mathsf{Enc}(\mathsf{ek}_1, \mathsf{tag}, x_1), \ldots, \mathsf{ct}_n \leftarrow \mathsf{Enc}(\mathsf{ek}_n, \mathsf{tag}, x_n))$ and n senders can independently contribute to the partial functional keys $\mathsf{dk}_{\mathbf{y},1} \leftarrow \mathsf{DKeyGen}(\mathsf{sk}_1, \mathsf{tag\text{-}f}, y_1), \ldots, \mathsf{dk}_{\mathbf{y},n} \leftarrow \mathsf{DKeyGen}(\mathsf{sk}_n, \mathsf{tag\text{-}f}, y_n)$ of some vector $\mathbf{y} = (y_1, \ldots, y_n)$. For the function class to compute inner products, many follow-up works improve upon the work of [24] on both aspects of efficiency as well as security, or by giving generic transformation to (D)MCFE from single-client FE [40, 2, 1]. All these works follow essentially the syntax of (D)MCFE in [24].

Function Privacy in FE. Standard security notions of all primitives mentioned above ensure that adversaries do not learn anything about the content of ciphertexts beyond what is revealed by the functions for which they possess decryption keys. However, it is *not* required that functional decryption keys hide the function they decrypt. In practice, this can pose a serious problem because the function itself could contain confidential data. For example, the evaluated function may represent an artificial neural network. Training such networks is often time-consuming and expensive, which is why companies offer their use as a paid service. However, to ensure that customers continue to pay for the use of the product, it is crucial that the concrete parameters of the network (*i.e.* the computed function) remain secret. This additional security requirement for functional encryption schemes is known as the so-called *function-hiding* property.
In particular, function-hiding functional encryption schemes for restricted function classes (such as inner products) have proven to be an important technical building block for the construction

[1] In contrast, MIFE involves no tags and thus a large amount of information can be obtained by arbitrarily combining ciphertexts to decrypt under some functional decryption key.

of functional encryption schemes for broader function classes: Lin [41] employed a function-hiding FE scheme for inner products to obtain a FE scheme for quadratic functions. A different technique was also introduced by Gay in [32] equally aiming at constructing FE for quadratic functions. With several technical novelties, Agrawal *et al.* [8, 10] were able to generalize the aforementioned constructions to obtain MIFE for quadratic functions.

1.1 Related Works

Generalizations of DMCFE In [26], Chotard *et al.* generalized DMCFE and defined the notion of *Dynamic Decentralized Functional Encryption* (DDFE) that allows participants to join at various stages during the lifetime of a system, while maintaining all decentralized features of DMCFE. The setup of DDFE can be non-interactive and decentralized, while that of DMCFE is interactive. The authors of [26] provided a concrete construction for DDFE for the function class computing inner products, which is provably secure in an indistinguishability-based model where the challenges must be submitted *selectively* and can handle only *static* corruption of private keys.
Besides, more multi-user FE primitives have been defined, such as *ad hoc multi-input functional encryption* [7] and *multi-authority attribute-based encryption* [22]. Interestingly, Agrawal *et al.* [9] recently proposed the very general notion of *Multi-Party Functional Encryption* (MPFE). The important concept behind MPFE is to cover all existing notions of FE in the multi-user setting, including DDFE/(D)MCFE. Moreover, as pointed out in [9], DDFE/(D)MCFE as presented in prior works *cannot* provide function-hiding. With delicate abstractions, MPFE captures the possibility of specifying public and private inputs for both ciphertext along with function keys, thus making it feasible to express function-hiding. In [9], Agrawal *et al.* proposed a construction for *function-hiding* DDFE, by specializing the syntax of MPFE, for the function class of inner products. Their scheme is provably secure in an indistinguishability-based model where the *single* challenge ciphertext and challenge key must be submitted *selectively* and can handle only *static* corruption of private keys.

Enhancements of FE with Function-Hiding Bishop *et al.* [17] presented the first FE scheme that guaranteed a weak variant of the function-hiding property. Shortly afterwards, the construction was lifted to fully function-hiding security by Datta *et al.* [28, 29]. This was further improved in terms of efficiency and/or computational hardness assumptions by works of Tomida *et al.* [50], Kim *et al.* [37] and Kim *et al.* [38]. The constructions of [17, 28, 50] all leverage the power of *dual pairing vector spaces* (DPVSes) developed by Okamoto and Takashima in [44, 45, 46].
In 2017, Lin [41] used a somewhat different approach that led to much simpler constructions. Roughly, she employs two instances of the public-key inner product FE scheme from DDH by Abdalla *et al.* [3] to hide both messages and keys at the same time. Using pairings, it is possible to decrypt both "layers" of ABDP encryption simultaneously and to produce exactly an encoding of the output inner product. This approach immediately generalizes to MIFE schemes, but it requires multilinear maps. Using the same blueprint but exploiting the specific algebraic properties of the MIFE scheme more carefully, Abdalla *et al.* [4] were able to construct function-hiding MIFE from standard bilinear maps. As mentioned earlier, Agrawal *et al.* [9] came up with the first construction of function-hiding MCFE for inner products which is inspired by the function-hiding MIFE scheme for inner products by Datta *et al.* [30]. Following the approach of Chotard *et al.* [26], they are then able to lift that scheme to function-hiding inner-product DDFE.

1.2 Our Contributions

Given the current state of the art, to the best of our knowledge, the only known (D)MCFE candidate that supports function-hiding comes from [9]. However, their FH-DDFE yields a construction for

function-hiding DMCFE (FH-DMCFE) in an implicit manner. In this paper, we investigate the problem of constructing *directly* FH-DMCFE for the function class of inner products, taking into account previous (non function-hiding) DMCFE in [24, 40, 2, 1] as well as the more general DDFE in [26, 9]. We aim at improving the resultant FH-DMCFE from the FH-DDFE of [9]:

1. We revisit the notion of FH-DMCFE and provide a new syntax, which is adapted from the one employed in previous works, together with a detailed function-hiding indistinguishability-based security model. A technical overview is given in Section 3.1 and the formal definitions are presented in Section 4.
2. For the function class computing inner products, we present candidates for FH-DMCFE based on pairings. Our constructions are provably secure in the ROM against *multiple adaptive* challenge encryption queries and *multiple adaptive* challenge key-generation queries, under *static* corruption. An overview highlighting our main technical ideas and the details of our achieved security level are given in Section 3.2, while the main constructions are presented in Section 6.
3. Our main technical contribution for achieving *adaptive* security on challenge ciphertexts and challenge keys is Lemma 8, which is proven and can find other applications in the DPVS setting. This lemma allows us to swap two coordinates of a vector in a basis, which leads to a *local* change in its product with *some* target vector in the dual basis, as long as we do not violate a *global* condition that binds the mentioned vector to *many* others.

Table 1 compares our candidates with existing works.

Scheme	**Type**	**Challenge**		**Corr.**	**Incompl.**[†]
		ciph	key		
[9, Section 6.2]	FH-MCFE	sing, sel	mult, sel	stat	✗
[9, Section 6.3]	FH-DDFE	sing, sel	mult, sel	stat	✗
Section 6.2	FH-DMCFE	mult, adap	mult, adap	stat	✗
Section 6.3	FH-DMCFE	mult, adap	mult, adap	stat	✓

[†] Tolerating these incomplete queries yields a stronger security model, *e.g.* see [25] for more details.

Table 1: We compare our constructions with existing works, in terms of the type of primitives (column **Type**), the number of allowed challenges and whether they can be adaptively queried (column **Chall.**, including both ciphertexts and keys as we are in function-hiding), the corruption model (column **Corr.**), and whether the adversary *is allowed to omit* some honest components in challenge queries (✓) or not (✗) (column **Incompl.**, see condition 1 in the technical overview). All schemes are defined for the function class $\mathcal{F}^{\mathsf{IP}} = \{F_{\mathbf{y}} : \mathbb{Z}_q^n \to \mathbb{Z}_q; \mathbf{x} \mapsto \langle \mathbf{x}, \mathbf{y} \rangle \in \mathcal{R}(\mathbb{Z}_q)\}$ where $n, q \in \mathbb{N}$, q is prime and $|\mathcal{R}(\mathbb{Z}_q)| = \mathsf{poly}(\log q)$. The shorthands $(\mathsf{mult}, \mathsf{sing}, \mathsf{sel}, \mathsf{adap}, \mathsf{dyn}, \mathsf{stat})$ denote multiple challenges, single challenges, selective challenges, adaptive challenges, dynamic corruption, static corruption.

2 Preliminaries

We write $[n]$ to denote the set $\{1, 2, \ldots, n\}$ 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 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 [31] and use $[\![a]\!]$ 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 $(\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_i, \ldots, \mathsf{G}_L)$ indexed by $i \in \{0, 1, \ldots, L\}$, we refer to the predecessor of G_j by 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*

$$D_0 = \{([\![1]\!], [\![a]\!], [\![b]\!], [\![ab]\!])\} \qquad D_1 = \{([\![1]\!], [\![a]\!], [\![b]\!], [\![c]\!])\}.$$

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

Definition 2. *In the bilinear setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$, the* **Symmetric eXternal Diffie-Hellman** (SXDH) *assumption makes the* DDH *assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

2.2 Dual Pairing Vector Spaces

Our constructions rely on the *Dual Pairing Vector Spaces* (DPVS) framework in prime-order bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$ and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}$ are all written additively. The DPVS technique dates back to the seminal work by Okamoto-Takashima [44, 45, 46] aiming at adaptive security for ABE as well as IBE, together with the *dual system methodology* introduced by Waters [51]. In [39], the setting for dual systems is composite-order bilinear groups. Continuing on this line of works, Chen *et al.* [23] 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 $\mathbf{x} = [\![(x_1, \ldots, x_N)]\!]_1 \in \mathbb{G}_1^N$ is identified as the vector $(x_1, \ldots, x_N) \in \mathbb{Z}_q^N$. There is no ambiguity because $\mathbb{G}_1$ is a cyclic group of order q prime. The $\mathbf{0}$-vector is $\mathbf{0} = [\![(0, \ldots, 0)]\!]_1$. 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 $t \cdot \mathbf{x} \coloneqq [\![t \cdot (x_1, \ldots, x_N)]\!]_1$, where $t \in \mathbb{Z}_q$ and $\mathbf{x} = [\![(x_1, \ldots, x_N)]\!]_1$. The additive inverse of $\mathbf{x} \in \mathbb{G}_1^N$ is defined to be $-\mathbf{x} \coloneqq [\![(-x_1, \ldots, -x_N)]\!]_1$. Viewing $\mathbb{Z}_q^N$ as a vector space of dimension N over $\mathbb{Z}_q$ with the notions of bases, we can obtain naturally a similar notion of bases for $\mathbb{G}_1^N$. More specifically, any invertible matrix $B \in GL_N(\mathbb{Z}_q)$ identifies a basis $\mathbf{B}$ of $\mathbb{G}_1^N$, whose i-th row $\mathbf{b}_i$ is $[\![B^{(i)}]\!]_1$, where $B^{(i)}$ is the i-th row of B. The canonical basis $\mathbf{A}$ of $\mathbb{G}_1^N$ consists of $\mathbf{a}_1 \coloneqq [\![(1, 0 \ldots, 0)]\!]_1, \mathbf{a}_2 \coloneqq [\![(0, 1, 0 \ldots, 0)]\!]_1, \ldots, \mathbf{a}_N \coloneqq [\![(0, \ldots, 0, 1)]\!]_1$. 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 $B \in GL_N(\mathbb{Z}_q)$. We write $\mathbf{x} = (x_1, \ldots, x_N)_\mathbf{B}$ to indicate the representation of $\mathbf{x}$ in the basis $\mathbf{B}$, i.e. $\mathbf{x} = \sum_{i=1}^N x_i \cdot \mathbf{b}_i$. By convention the writing $\mathbf{x} = (x_1, \ldots, x_N)$ concerns the canonical basis $\mathbf{A}$. For the conciseness at some point when we focus on the indices in an ordered list L of length ℓ, we write $\mathbf{x} = (x_{L[1]}, \ldots, x_{L[\ell]})_{\mathbf{B}[L]}$. Treating $\mathbb{G}_2^N$ similarly, we can furthermore define a product of two vectors $\mathbf{x} = [\![(x_1, \ldots, x_N)]\!]_1 \in \mathbb{G}_1^N, \mathbf{y} = [\![(y_1, \ldots, y_N)]\!]_2 \in \mathbb{G}_2^N$ by $\mathbf{x} \times \mathbf{y} \coloneqq \prod_{i=1}^N \mathbf{e}(\mathbf{x}[i], \mathbf{y}[i]) = [\![\langle (x_1, \ldots, x_N), (y_1, \ldots, y_N) \rangle]\!]_\mathsf{t}$. Given a basis $\mathbf{B} = (\mathbf{b}_i)_{i \in [N]}$ of $\mathbb{G}_1^N$, we define $\mathbf{B}^*$ to be a basis of $\mathbb{G}_2^N$ by first defining $B' \coloneqq (B^{-1})^\top$ and the i-th row $\mathbf{b}_i^*$ of $\mathbf{B}^*$ is $[\![B'^{(i)}]\!]_2$. It holds that $B \cdot (B')^\top = I_N$ the identity matrix and $\mathbf{b}_i \times \mathbf{b}_j^* = [\![\delta_{i,j}]\!]_\mathsf{t}$ for every $i, j \in [N]$, where

$\delta_{i,j} = 1$ if and only if $i = j$. We call the pair $(\mathbf{B}, \mathbf{B}^*)$ a *pair of dual orthogonal bases* of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$. If $\mathbf{B}$ is constructed by a random invertible matrix $B \xleftarrow{\$} GL_N(\mathbb{Z}_q)$, we call the resulting $(\mathbf{B}, \mathbf{B}^*)$ a pair of random dual bases. A DPVS is a bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q, N)$ with dual orthogonal bases. In this work, we also use extensively *basis changes* over dual orthogonal bases of a DPVS to argue the steps of switching key as well as ciphertext vectors to semi-functional mode in our proofs. The details of such basis changes are recalled in Appendix A.5.

3 Technical Overview

3.1 Syntax and Security Notions for FH-DMCFE

Syntax. In [24] and its follow-up works [25, 2, 1, 40, 26] on DMCFE, the syntax of DMCFE is given by five algorithms (Setup, Enc, DKeyGen, DKeyComb, Dec) where Setup allows n clients and n senders initialize and agree upon their encryption keys $(\mathsf{ek}_i)_i$ as well as their secret keys $(\mathsf{sk}_i)_i$, respectively[2]. Each client can perform an encryption $\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i) \to \mathsf{ct}_i$ on their private component x_i using their private ek_i, under some tag tag. Each sender can independently generate a partial functional key $\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, F) \to \mathsf{dk}_{F,i}$ using their private sk_i on a common function F, under some tag tag-f. These partial functional keys can be combined by $\mathsf{DKeyComb}((\mathsf{dk}_{F,i})_i, \mathsf{tag\text{-}f}) \to \mathsf{dk}_F$. The decryption is run on $(\mathsf{ct}_i)_i$ using the combined key dk_F. All these prior works do not consider *function-hiding* property for DMCFE and thus the syntax therein, for which the *complete* function F is given at the time of partial key generation, needs certain refinements to be able to capture the privacy of functions.

Very recently, Agrawal *et al.* introduced MPFE in [9] so as to cover all existing notions of FE in the multi-user setting, including DMCFE. Up to this purpose, the syntax of MPFE presented in [9] can be specialized to describe DMCFE schemes. However, when translating MPFE into DMCFE, for example see [9, Section 4], the induced syntax also passes the *complete* description of F in the public part (coined "$y_{pub,i}$" in [9] for each sender i) of the argument to the key-generation algorithm. Thus it falls back on the syntactical problem we mentioned above. We emphasize that later in [9, Section 6.3], Agrawal *et al.* constructed a *function-hiding* DDFE scheme for inner products that includes FH-DMCFE as a particular case but their security level as an FH-DMCFE is against one *selective challenge ciphertext* under *static* corruption.

Therefore, our starting point is to devise a concrete syntax for DMCFE that can formalize the function-hiding property. For simplicity, we do not consider the algorithm DKeyComb for combining partial functional keys into a fully functional key. In the work of [24], this algorithm DKeyComb is merely for efficiency enhancement. The decryption by Dec now receives all partial keys $(\mathsf{dk}_i)_i$ in order to decrypt the ciphertext components $(\mathsf{ct}_i)_i$. The syntax for Enc stays the same. Concerning DKeyGen, it is more appropriate to formalize function-hiding if each sender i does *not* receive the complete description of F by default, but only a piece of information that, together with other senders, can be used to determine F. For this reason, we encode the function F using n parameters $(y_1, \ldots, y_n)$, and for the generation of the i-th partial key of the i-th sender, only y_i is needed. We implicitly use a deterministic encoding to encode F into $(y_1, \ldots, y_n)$[3]. Last but not least, the tag tag-f to be used at the time of DKeyGen contains only generic public information of F, *e.g.* its purpose, and not its specific parameters. We refer to Section 4 for the formal definitions.

[2] This setup procedure is centralized in MCFE and interactive in DMCFE.

[3] The description of such an encoding might very well depend on the functionality that is under consideration. For instance, in the case of inner products $F_\mathbf{y} : (\mathbb{Z}_q^*)^n \to \mathbb{Z}_q$ that is defined as $F_\mathbf{y}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle$, the n parameters of $F_\mathbf{y}$ is simply $\mathbf{y}$.

Security. Intuitively, security requires that all ppt adversaries cannot guess a randomly chosen bit $b \xleftarrow{\$} \{0,1\}$ with non-negligible probability in the following game: initially, the adversary receives the public parameters of the system. It then can adaptively query three oracles: a left-or-right encryption oracle $\mathcal{O}\mathsf{Enc}$, a left-or-right key-generation oracle $\mathcal{O}\mathsf{KeyGen}$, and a corruption oracle $\mathcal{O}\mathsf{Corrupt}$ which provides encryption *and* secret keys $(\mathsf{ek}_i, \mathsf{sk}_i)$ for any i of its choice. This is corresponding to the original corruption model in [24] and to the best of our knowledge, other works on the subject of DMCFE either identify $\mathsf{sk}_i = \mathsf{ek}_i$, *e.g.* [1], or sk_i contains ek_i, *e.g.* [40]. Therefore corrupting one type of keys has impacts on the other. Finally, the adversary submits its guess b' for b.
Given the security game defined as above, we cannot yet prove function-hiding security for any DMCFE schemes, due to the fact that there are attacks that help the adversary trivially win the game. This problem is fundamental to FE in general and to (D)MCFE in particular, as put forth by Chotard *et al.* in [24]. Therefore, we must restrict what kind of queries an adversary can ask during the security game, and this restriction is checked in the Finalize procedure of the game[4]. Most importantly, all complete ciphertexts of messages $(x_1^b, \ldots, x_n^b)$ and complete decryption keys for functions F^b that the adversary obtained via $(\mathcal{O}\mathsf{Enc}, \mathcal{O}\mathsf{KeyGen})$ must satisfy

$$F^0(x_1^0, \ldots, x_n^0) = F^1(x_1^1, \ldots, x_n^1) \ . \tag{1}$$

The ensemble of conditions to be checked during Finalize will determine if an adversary is *admissible* or not, and its guess b' is taken into account only if it is admissible. We refer to Definition 4 for the formal definition of function-hiding security for DMCFE.

3.2 Function-Hiding DMCFE for Inner Products

After defining the function-hiding security for DMCFE in Section 4, with various levels of security and relations among them (see Lemma 6 and Lemma 7), we turn our attention to FH-DMCFE for the function class computing inner products. As mentioned in Section 3.1, our goal is to improve upon the DDFE for inner products in [9, Section 6.3], when being viewed as a DMCFE. Our result is twofold. We first give an intermediate construction in Section 6.1 that is *function-hiding* against *one adaptive* challenge, under *static* corruption and the following constraint on the adversary's queries:

1. *(Complete queries contraint - Condition 1 in Definition 4)* If there exists a challenge ciphertext (or key) query for any honest component, then all honest challenge ciphertext (or key) components must be queried.

Afterwards, we can apply similar techniques introduced in [42] and [4] to the construction in Section 6.1 and make it function hiding against multiple adaptive challenges.
Later, in Section 6.3, we give a generic transformation that makes our construction from Section 6.1 an FH-DMCFE against *multiple adaptive* ciphertext challenges, under *static* corruption but *without* the preceding constraint. Similarly, we also first treat the single-challenge case then transform it into a fully function-hiding DMCFE. Given below are the high-level ideas of our construction.

Construction based on SXDH. Our construction relies on the notion of *Dual Pairing Vector Spaces* (DPVSes, see Section 2.2). In the following we highlight the main ideas of our backbone construction in Section 6.1. Our function class of interest is for computing inner products $\mathcal{F}^{\mathsf{IP}} = \{F_{\mathbf{y}}\}$ where $F_{\mathbf{y}} : (\mathbb{Z}_q^*)^n \to \mathbb{Z}_q$ is defined as $F_{\mathbf{y}}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle$. The parameter vector of $F_{\mathbf{y}}$ is simply $\mathbf{y} = (y_1, \ldots, y_n) \in \mathbb{Z}_q^n$. We use DPVSes in the bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathsf{t}}, \mathbf{e}, g_1, g_2, g_{\mathsf{t}})$. We use two hash functions $\mathsf{H}_1 : \mathsf{Tag} \to \mathbb{G}_1$ and $\mathsf{H}_2 : \mathsf{Tag} \to \mathbb{G}_2$ to process the encryption and key

[4] For general functionality, the restriction might not be efficiently decidable, but in our concrete schemes for inner products, it is indeed the case.

generation tags. Given tag for encryption and $\mathsf{tag\text{-}f}$ for key generation, we denote $[\![\omega]\!]_1 \leftarrow \mathsf{H}_1(\mathsf{tag})$ and $[\![\mu]\!]_2 \leftarrow \mathsf{H}_2(\mathsf{tag\text{-}f})$. We employ two sets of secret sharings of 0, namely $(s_i)_i$ in the key and $(t_i)_i$ in the ciphertext, so that they will cancel when all keys of the same $\mathsf{tag\text{-}f}$ and all ciphertexts of the same tag are combined. This can be done by using $[\![\omega]\!]_1$ to randomize a secret sharing $(\tilde{t}_i)_i$ of 0, which is generated at setup and embedded in the encryption keys ek_i, so as to obtain $[\![t_i]\!]_1 := [\![\omega\tilde{t}_i]\!]_1$. The same technique is done for the decentralized generation of $(s_i)_i := (\mu\tilde{s}_i)_i$, *i.e.* by using μ to randomize a pre-generated secret sharing $(\tilde{s}_i)_i$. Additionally, we need more coordinates for the goal of *multiple adaptive* challenge ciphertexts and challenge keys. More specifically, the ciphertext components $\mathbf{c}_i$ and the partial key components $\mathbf{d}_i$ are as follows:

$$\begin{array}{rccc|c|c|c|c|c|l} \mathbf{c}_i = (& x_i & \omega & 0 & 0 & 0 & 0 & t_i &)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& y_i & s_i & 0 & 0 & 0 & 0 & \mu &)_{\mathbf{B}_i^*} \end{array}$$

Turning to its security, we first notice that Lemma 6 proves that in the *weakly function-hiding* setting[5] and *static* corruption, security against *one* challenge is equivalent to security against *multiple* challenges; Furthermore, Lemma 7 proves that under *static* corruption, an $\mathsf{FH\text{-}DMCFE}$ scheme that is *weakly function-hiding* (against multiple challenges) can be made *function-hiding* with a constant loss in efficiency and security. We thus can focus on proving the security of our scheme in the *(weakly) function-hiding one-challenge* setting, while the challenge (key and ciphertext) can be *adaptively* queried, and the corruption is *static*.

We emphasize that our one-challenge security notion means there exists only one tag^* to $\mathcal{O}\mathsf{Enc}$ having $(x_i^0)_i \neq (x_i^1)_i$, while for other $\mathsf{tag}_\ell \neq \mathsf{tag}^*$ it holds that $(x_i^0)_{\ell,i} = (x_{\ell,i}^1)_i$. We denote by $(\mathbf{c}_{\ell,i})_i$ the ciphertext components for $(x_{\ell,i})_i$ under these non-challenge tag_ℓ. In the same manner, there exists only one $\mathsf{tag\text{-}f}^*$ to $\mathcal{O}\mathsf{KeyGen}$ having $(y_i^0)_i \neq (y_i^1)_i$, while for other $\mathsf{tag\text{-}f}_k \neq \mathsf{tag\text{-}f}^*$ it holds that $(y_i^0)_{k,i} = (y_{k,i}^1)_i$. We denote by $(\mathbf{d}_{k,i})_i$ the key components for $(y_{k,i})_i$ under these non-challenge $\mathsf{tag\text{-}f}_k$. To summarize, starting from the game with the challenge bit $b = 0$, the information that an adversary will obtain consists of:

$$\begin{array}{rccc|c|c|c|c|c|l} \mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & t_{\ell,i} &)_{\mathbf{B}_i} \\ \mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & 0 & 0 & 0 & 0 & \mu_k &)_{\mathbf{B}_i^*} \\ \mathbf{c}_i = (& x_i^0 & \omega & 0 & 0 & 0 & 0 & t_i &)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& y_i^0 & s_i & 0 & 0 & 0 & 0 & \mu &)_{\mathbf{B}_i^*} \end{array} .$$

The challenge $(\mathbf{c}_i, \mathbf{d}_i)_i$ are queried adaptively, and the set of corrupted i is declared up front. Our goal is to switch $(x_i^0, y_i^0)_i$ in $(\mathbf{c}_i, \mathbf{d}_i)_i$ to $(x_i^1, y_i^1)_i$.

The key technique in our proof is using basis changes in DPVS, where for each i, the dual bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are used to express the ciphertexts $(\mathbf{c}_i, \mathbf{c}_{\ell,i})$ and the keys $(\mathbf{d}_i, \mathbf{d}_{k,i})$, respectively. Because we know in advance which i is corrupted, whose keys $(\mathsf{ek}_i, \mathsf{sk}_i)$ will be revealed to the adversary, the basis changes are applied only to the *honest* i whose keys are never revealed. This is indeed possible, *i.e.* we can focus on the honest components and are able to switch $(x_i^0, y_i^0)_i$ in $(\mathbf{c}_i, \mathbf{d}_i)_i$ to $(x_i^1, y_i^1)_i$, because the admissibility following Definition 4 dictates that in the case of inner products, we have $x_i^0 = x_i^1$ and $y_i^0 = y_i^1$ for all *corrupted* i. We perform the switching in multiple steps, the changes are indicated by boxed components.

Pre-processing Our idea for changing $(x_i^0, y_i^0)_i$ to $(x_i^1, y_i^1)_i$ for honest i is applying basis changes on the coordinates $(1, 2)$ of $(\mathbf{B}_i, \mathbf{B}_i^*)$ to modify $(\mathbf{c}_i, \mathbf{d}_i)$. However, this will have affects on $(\mathbf{c}_{\ell,i}, \mathbf{d}_{k,i})$, *i.e.* the non-challenge vectors as well. The crucial point is that our simulation should always preserve the decryption not only between the challenge key and the challenge ciphertext, but also between the combination of challenge and non-challenge vectors. Therefore, as a pre-processing step, we isolate

[5] That is, we change the requirement (1) to $F^0(x_1^0, \ldots, x_n^0) = F^1(x_1^0, \ldots, x_n^0) = F^1(x_1^1, \ldots, x_n^1)$.

the values $(x_{\ell,i}, y_{k,i})$ as well as $(\omega, s_{k,i})$ in $(\mathbf{c}_{\ell,i}, \mathbf{d}_{k,i})$ so that the subsequent changes on $(\mathbf{B}_i, \mathbf{B}_i^*)$ targeting $(\mathbf{c}_i, \mathbf{d}_i)$ will not affect theses values:

$$\begin{array}{rccc|c|c|c|c|l} \mathbf{c}_{\ell,i} = (& 0 & 0 & \boxed{x_{\ell,i}} & 0 & \boxed{\omega_\ell} & 0 & t_{\ell,i} &)_{\mathbf{B}_i} \\ \mathbf{d}_{k,i} = (& 0 & 0 & \boxed{y_{k,i}} & \boxed{y_{k,i}} & \boxed{s_{k,i}} & \boxed{s_{k,i}} & \mu_k &)_{\mathbf{B}_i^*} \\ \mathbf{c}_i = (& x_i^0 & \omega & 0 & \boxed{x_i^0} & 0 & \boxed{\omega} & t_i &)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& y_i^0 & s_i & \boxed{y_i^0} & 0 & \boxed{s_i} & 0 & \mu &)_{\mathbf{B}_i^*} \end{array} .$$

This pre-processing step will rely on the SXDH assumption in $(\mathbb{G}_1, \mathbb{G}_2)$ and is computational.

Complexity leveraging under formal changes We arrive at a game in which only the challenge vectors $(\mathbf{c}_i, \mathbf{d}_i)$ have non-trivial values at coordinate $(1, 2)$. We now proceed to mod $(x_i^0, y_i^0)_i$ into $(x_i^1, y_i^1)_i$ for honest i. Our key observation is that: thanks to the *static* corruption, we know since the very beginning for which i the keys are never leaked, therefore we can perform *formal* basis changes for those $(\mathbf{B}_i, \mathbf{B}_i^*)$ that will guarantee a perfect transition from $(x_i^0, y_i^0)_i$ to $(x_i^1, y_i^1)_i$ in $(\mathbf{c}_i, \mathbf{d}_i)$. The fact that i is honest is crucial as such formal basis changes essentially modify the basis vectors, which can be recognized by the adversary if $(\mathsf{ek}_i, \mathsf{sk}_i)$ are corrupted by encrypting/generating on their own the ciphertext/key components, then performing products with the simulated challenge vectors. Intuitively, those basis changes will shift the basis vectors by a factor that depends on y_i^0, y_i^1, and $x_i^0 y_i^0 - x_i^1 y_i^1$.
A pitfall in the above idea is that all basis changes must be performed at setup time, *i.e.* the simulator cannot change the bases during the course of query-response of the game. Therefore, we first consider the *selective* version of the games where $(x_i^0, y_i^0)_i$ and $(x_i^1, y_i^1)_i$ are known in advance, then the formal basis changes (defined using those selective challenge ciphertext and key values) give us a perfectly indistinguishable transition from $(x_i^0, y_i^0)_i$ into $(x_i^1, y_i^1)_i$ for honest i[6]. Finally, we apply a *complexity leveraging* on these selective games, which preserves the perfect indistinguishability and arrive at:

$$\begin{array}{rccc|c|c|c|c|l} \mathbf{c}_{\ell,i} = (& 0 & 0 & x_{\ell,i} & 0 & \omega_\ell & 0 & t_{\ell,i} &)_{\mathbf{B}_i} \\ \mathbf{d}_{k,i} = (& 0 & 0 & y_{k,i} & y_{k,i} & s_{k,i} & s_{k,i} & \mu_k &)_{\mathbf{B}_i^*} \\ \mathbf{c}_i = (& \boxed{x_i^1} & \omega & 0 & x_i^0 & 0 & \omega & t_i &)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& \boxed{y_i^1} & \boxed{s_i'} & y_i^0 & 0 & s_i & 0 & \mu &)_{\mathbf{B}_i^*} \end{array} .$$

We remark that under these formal basis changes, the secret share s_i at coordinate 2 is modified into a new secret sharing $s_i' := s_i - (x_i^0 y_i^0 - x_i^1 y_i^1)/\omega$.

Post-processing After switching from $(x_i^0, y_i^0)_i$ to $(x_i^1, y_i^1)_i$ in $(\mathbf{c}_i, \mathbf{d}_i)$, at coordinates $(1, 2)$, we have to move $(x_{\ell,i}, y_{k,i})$ as well as $(\omega_\ell, s_{k,i})$ back to these positions to be correctly in the game whose challenge bit is $b = 1$. We exploit the *function-hiding* property, which implies that for all ℓ, k we have

$$\sum_{i=1}^n x_{\ell,i} y_i^0 = \sum_{i=1}^n x_{\ell,i} y_i^1 \quad \text{and} \quad \sum_{i=1}^n x_i^0 y_{k,i} = \sum_{i=1}^n x_i^1 y_{k,i} .$$

This implies

$$\sum_{\text{honest } i} x_{\ell,i} y_i^0 = \sum_{\text{honest } i} x_{\ell,i} y_i^1 \quad \text{and} \quad \sum_{\text{honest } i} x_i^0 y_{k,i} = \sum_{\text{honest } i} x_i^1 y_{k,i}$$

due to the constraint that $x_i^0 = x_i^1$ and $y_i^0 = y_i^1$ for all *corrupted* i. As a byproduct from the above observation, the new secret shares s_i' for honest i resulted from the switching also satisfies

[6] We recall that there is nothing to be done concerning the corrupted i due to the admissibility.

$\sum_{\text{honest } i} s'_i = \sum_{\text{honest } i} s_i$. Thus, after a preparatory swapping (feasible using DDH as $\mathbf{c}_i[2] = \mathbf{c}_i[6]$ and $\mathbf{c}_{\ell,i}[2] = \mathbf{c}_{\ell,i}[6]$):

$$\begin{array}{rccc|c|c|c|c|c|l}
\mathbf{c}_{\ell,i} = (& 0 & 0 & x_{\ell,i} & 0 & \omega_\ell & 0 & t_{\ell,i} &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& 0 & \boxed{s_{k,i}} & y_{k,i} & y_{k,i} & s_{k,i} & \boxed{0} & \mu_k &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & t_i &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s'_i & y_i^0 & 0 & s_i & 0 & \mu &)_{\mathbf{B}_i^*}
\end{array}$$

However, if we move $x_{\ell,i}$ back to coordinate 1 in $\mathbf{c}_{\ell,i}$, then *globally* there is no difference because $\sum_{\text{honest } i} x_{\ell,i} y_i^0 = \sum_{\text{honest } i} x_{\ell,i} y_i^1$ but the *local* term is changed from $x_{\ell,i} y_i^0$ to $x_{\ell,i} y_i^1$, for $\mathbf{d}_i[1] \neq \mathbf{d}_i[3]$. We develop a technical tool to deal with this situation, where we need to conduct local perturbations without impacting the vectors' global relation, called *secret-sharing swapping*. Using this tool in conjunction with the current *static* corruption setting, we can handle in parallel such perturbations for all honest i and successfully move back $(x_{\ell,i}, y_{k,i})$ as well as $(\omega_\ell, s_{k,i})$. At the end a cleaning is needed to formally make the vectors conform to the security game for $b = 1$:

$$\begin{array}{rcc|c|c|c|c|c|l}
\mathbf{c}_{\ell,i} = (& \boxed{x_{\ell,i}} & \boxed{\omega_\ell} & 0 & 0 & 0 & 0 & t_{\ell,i} &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& \boxed{y_{k,i}} & \boxed{s_{k,i}} & 0 & 0 & 0 & 0 & \mu_k &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & 0 & 0 & 0 & t_i &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s_i & 0 & 0 & 0 & 0 & \mu &)_{\mathbf{B}_i^*}
\end{array}$$

and the proof is completed. Section 5 states and proves Lemma 8 for secret-sharing swappping. For our FH-DMCFE, we refer to Section 6.1 for the construction and Theorem 9 for the proof.

Achieving security against incomplete queries. In Section 6.3, we enhance our constructions from Section 6.1 so as to relax the complete queries constraint 1. All over again, we can focus on first building DMCFE schemes that achieve *(weakly) function-hiding* property against *one adaptive challenge ciphertext and challenge key*, under *static* corruption, then applying Lemma 6 and Lemma 7. Our main idea naturally extends the work by Abdalla *et al.* [1] and uses *pseudorandom functions* (PRF) together with *IND-CPA secure symmetric encryption* (SE) in order to transform a DMCFE that is secure under the complete queries constraint 1 and remove this constraint. The setup now generates $2n^2$ keys for each pair $(i, j) \in [n] \times [n]$. Each group of n^2 keys are used for the ciphertext components and key components, respectively. For instance, when generating the i-th ciphertext component for (x_i^0, x_i^1) under tag, the underlying DMCFE's component ct_i is encrypted using the SE while the key being the XOR-ing of all PRF evaluations on tag using the keys w.r.t (i, j) for $j \in [n]$. The ciphertext also contains the "expected contribution" of i in other j-th components, *i.e.* the PRF evaluations on tag using the keys w.r.t (j, i). This increases the ciphertext size by an $\Theta(n)$ factor and the total communication is now $\Theta(n^2)$. Intuitively, for any j, only when *all* i-th ciphertext components are obtained, can the key for SE decryption be computed (as all pairs (j, i) are now present) and allow SE decryption to ct_i. Otherwise, we employs the PRF security on the missing i-th component to switch the SE key to a uniformly random, then apply its IND-CPA security so as to replace x_i^0 by x_i^1. This can be done in the *static* corruption setting in which we know in advance for which honest i whose $(\mathsf{ek}_i, \mathsf{sk}_i)$ are not revealed and is omitted in the adversary's challenge queries. We also need the weakly function-hiding property as during our hybrids there are mixings of left- and right-challenge (key with ciphertext) queries. In the end, only when *all* i-th ciphertext components are obtained can we perform the SE decryption to obtain the $(\mathsf{ct}_i)_{i \in [n]}$ of the underlying DMCFE. We refer to Section 6.3 for more details.

4 Function-Hiding Decentralized Multi-Client FE

We introduce the notion of *function-hiding decentralized multi-client functional encryption* (FH-DMCFE).

Definition 3 (Decentralized Multi-Client Functional Encryption). *Let $\lambda \in \mathbb{N}$ and $n = n(\lambda) : \mathbb{N} \to \mathbb{N}$ be a function. Let $\mathcal{F} = \{f : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathcal{R}\}$ be a function family, where each $f \in \mathcal{F}$ is defined by n parameters in $\mathsf{Param}_1 \times \cdots \times \mathsf{Param}_n$[7]. Furthermore, let Tag denote a set of tags used for ciphertext and function components. A* decentralized multi-client functional encryption (DMCFE) scheme *$\mathcal{E}$ for $\mathcal{F}$ between n senders $(\mathcal{S}_i)_{i\in[n]}$ and a functional decrypter $\mathcal{FD}$ consists of the four algorithms $(\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ defined below:*

- $\mathsf{Setup}(1^\lambda)$**:** *This is a protocol between the senders $(\mathcal{S}_i)_{i\in[n]}$ that eventually generate their own secret keys sk_i and encryption keys ek_i, as well as the public parameter pp. We will assume that all the secret and encryption keys implicitly contain pp.*
- $\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i)$**:** *On input a user secret key sk_i, a tag $\mathsf{tag\text{-}f} \in \mathsf{Tag}$, and parameter $y_i \in \mathsf{Param}_i$, this algorithm outputs a partial functional decryption key $\mathsf{dk}_{\mathsf{tag\text{-}f},i}$.*
- $\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i)$**:** *On input an encryption key ek_i, a tag tag and a message $x_i \in \mathcal{D}_i$, this algorithm outputs a ciphertext $\mathsf{ct}_{\mathsf{tag},i}$.*
- $\mathsf{Dec}(\mathbf{d}, \mathbf{c})$**:** *On input a list of functional decryption keys $\mathbf{d} := (\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i=1}^n$ and a list of ciphertexts $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i})_{i=1}^n$, this algorithm outputs an element $d \in \mathcal{R}$ or a symbol $\perp$.*

For efficiency, prior papers (such as [24, 25, 2, 1, 40, 26]) considered an additional fifth algorithm $\mathsf{DKeyComb}((\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i\in[n]})$ that, given n partial decryption keys $(\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i\in[n]}$ generated for the same tag $\mathsf{tag\text{-}f}$, outputs a succinct functional decryption key $\mathsf{dk}_{\mathsf{tag\text{-}f}}$ which can be passed to $\mathsf{Dec}(\mathsf{dk}_{\mathsf{tag\text{-}f}}, \mathbf{c})$. Also, the algorithm $\mathsf{DKeyGen}$ (called $\mathsf{DKeyGenShare}$ in [24]) usually receives only two arguments, a secret key sk_i and a tag $\mathsf{tag\text{-}f}$ containing a description of the corresponding function. However, in the context of function-hiding DMCFE, we consider it more appropriate if each party does not receive the complete description of the function by default, but only the part of the description necessary for the computation of its partial decryption key. For this reason, we decompose the description of a function into n parameters $(y_1, \ldots, y_n) \in \mathsf{Param}_1 \times \cdots \times \mathsf{Param}_n$ while y_i contains only the information necessary for the computation of the i-th partial decryption key $\mathsf{dk}_{\mathsf{tag\text{-}f},i}$. On the other hand, the tag $\mathsf{tag\text{-}f}$ is intended to only include the generic public purpose of the function.

Correctness. $\mathcal{E}$ is *correct* if for all $\lambda \in \mathbb{N}$, $(x_1, \ldots, x_n) \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$, $f \in \mathcal{F}$ having parameters $(y_i)_{i=1}^n \in \mathsf{Param}_1 \times \cdots \times \mathsf{Param}_n$, and any $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$, we have

$$\Pr\left[d = f(x_1, \ldots, x_n) \;\middle|\; \begin{array}{l} (\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i) \\ \mathsf{dk}_{\mathsf{tag\text{-}f},i} \leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i) \\ d := \mathsf{Dec}((\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i\in[n]}, (\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]}) \end{array} \right] = 1$$

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

Security. We define the *function-hiding* security for DMCFE. In the seminal work by Chotard *et al.* [24] and its follow-up study [26], the security notion does not cover the function-hiding requirement for DMCFE or its more general sibling *Dynamic Decentralized Functional Encryption* (DDFE). Until recently, the work by Agrawal *et al.* [9] abstracted out DMCFE into the notion of *Multi-Party Functional Encryption* (MPFE). The authors of [9] also used MPFE to spell out the function-hiding security for MCFE as well as for DDFE. The latter does capture DMCFE as a particular case but as our current work focuses on DMCFE, we introduce the detailed function-hiding security for DMCFE, without going through all the abstraction of MPFE nor of DDFE.

[7] Implicitly, we use a deterministic encoding $\mathsf{p} : \mathcal{F} \to \mathsf{Param}_1 \times \cdots \times \mathsf{Param}_n$ in order to associate each function to its parameters.

Definition 4 (Function-Hiding Security). *For a DMCFE scheme $\mathcal{E}$, a function class $\mathcal{F}$ and a* ppt *adversary $\mathcal{A}$ we define the experiment $\mathbf{Exp}^{\text{adap-dyn-fh-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ as shown in Figure 1. The oracles $\mathcal{O}\mathsf{Enc}$, $\mathcal{O}\mathsf{DKeyGen}$ and $\mathcal{O}\mathsf{Corrupt}$ can be called in any order and any number of times. The adversary $\mathcal{A}$ is* NOT admissible *with respect to $\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{DKGen}}$, denoted by $\mathsf{adm}(\mathcal{A}) = 0$, if either one of the following holds:*

1. *(Complete queries constraint) There exists $\mathsf{tag} \in \mathsf{Tag}$ so that $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, x_i^0, x_i^1)$ have been asked for some but not all $i \in \mathcal{H} := [n] \setminus \mathcal{C}$, or there exists $\mathsf{tag\text{-}f} \in \mathsf{Tag}$ such that $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1)$ have been asked for some but not all $i \in \mathcal{H} := [n] \setminus \mathcal{C}$.*
2. *There exist two distinct tuples of the form $(i, \mathsf{tag}, \cdot, \cdot)$ in $\mathcal{Q}_{\mathsf{Enc}}$ or two distinct tuples of the form $(i, \mathsf{tag\text{-}f}, \cdot, \cdot)$ in $\mathcal{Q}_{\mathsf{DKGen}}$.*
3. *There exists a tuple $(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ such that $i \in \mathcal{C}$ and $x_i^0 \neq x_i^1$, or there exists $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ such that $i \in \mathcal{C}$ and $y_i^0 \neq x_i^1$.*
4. *(Function-hiding) There exist $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$, two vectors $(x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]} \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ and two functions $f^0, f^1 \in \mathcal{F}$ having parameters $(y_i^0, y_i^1)_{i=1}^n$ such that*
   - *$(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ and $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ for all $i \in \mathcal{H}$,*
   - *$x_i^0 = x_i^1$ and $y_i^0 = y_i^1$ for all $i \in \mathcal{C}$, and*
   - *$f^0(x_1^0, \ldots, x_n^0) \neq f^1(x_1^1, \ldots, x_n^1)$.*

Otherwise, we say that $\mathcal{A}$ is admissible w.r.t $\mathcal{C}$, $\mathcal{Q}_{\mathsf{Enc}}$ and $\mathcal{Q}_{\mathsf{DKGen}}$ *and write $\mathsf{adm}(\mathcal{A}) = 1$. We call $\mathcal{E}$* function-hiding secure against adaptive challenges and dynamic corruption of clients *if for all* ppt *adversaries $\mathcal{A}$,*

$$\mathbf{Adv}^{\text{adap-dyn-fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left|\Pr\left[\mathbf{Exp}^{\text{adap-dyn-fh-0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{adap-dyn-fh-1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right]\right|$$

is negligible in λ.

$\underline{\mathsf{Initialize}(1^\lambda)}$:
$\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{DKGen}} \leftarrow \varnothing$
$(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda)$
Return pp

$\underline{\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, x_i^0, x_i^1)}$:
$\mathcal{Q}_{\mathsf{Enc}} \leftarrow \mathcal{Q}_{\mathsf{Enc}} \cup \{(i, \mathsf{tag}, x_i^0, x_i^1)\}$
Return $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i^b)$

$\underline{\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1)}$:
$\mathcal{Q}_{\mathsf{DKGen}} \leftarrow \mathcal{Q}_{\mathsf{DKGen}} \cup \{(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1)\}$
Return $\mathsf{dk}_{f,i} \leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i^b)$

$\underline{\mathcal{O}\mathsf{Corrupt}(i)}$:
$\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$; return $(\mathsf{sk}_i, \mathsf{ek}_i)$

$\underline{\mathsf{Finalize}(b')}$:
If $\mathsf{adm}(\mathcal{A}) = 1$, return $\beta \leftarrow b'$
Else, return $\beta \xleftarrow{\$} \{0, 1\}$

Fig. 1: Security game $\mathbf{Exp}^{\text{adap-dyn-fh-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ for Definition 4

Weaker notions. One may define weaker variants of indistinguishability by restricting the access to the oracles and imposing stronger admissibility conditions.

- *Security against Static Corruption:* The experiment $\mathbf{Exp}^{\text{adap-stat-fh-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\text{adap-dyn-fh-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that all queries to the oracle $\mathcal{O}\mathsf{Corrupt}$ must be submitted before $\mathsf{Initialize}$ is called.

- *Security against Selective Challenges:* The experiment $\mathbf{Exp}^{\mathsf{sel\text{-}dyn\text{-}fh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{adap\text{-}dyn\text{-}fh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that all queries to the oracle $\mathcal{O}\mathsf{Enc}$ must be submitted before $\mathsf{Initialize}$ is called.
- *One-time Security:* The experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}dyn\text{-}fh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{adap\text{-}dyn\text{-}fh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that the adversary must declare up front to $\mathsf{Initialize}$ two additional "challenge" tags $\mathsf{tag}^*, \mathsf{tag\text{-}f}^* \in \mathsf{Tag}$ such that for all $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$:
  - if $(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ and $\mathsf{tag} \neq \mathsf{tag}^*$, then $x_i^0 = x_i^1$,
  - if $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ and $\mathsf{tag\text{-}f} \neq \mathsf{tag\text{-}f}^*$, then $y_i^0 = y_i^1$.
- *Weakly Function-Hiding:* We can weaken the function-hiding property by changing condition 4 for $\mathsf{adm}(\mathcal{A}) = 0$. More specifically, we replace it by the following condition *4'*:

  4'. (Weakly Function-hiding) *There exist* $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$, $(x_i^0)_{i\in[n]}$ *and* $(x_i^1)_{i\in[n]}$ *in* $\mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ *and two functions* $f^0, f^1 \in \mathcal{F}$ *having parameters* $(y_i^0, y_i^1)_{i=1}^n$ *such that*
  - $(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ *and* $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ *for all* $i \in \mathcal{H}$,
  - $x_i^0 = x_i^1$ *and* $y_i^0 = y_i^1$ *for all* $i \in \mathcal{C}$, *and*
  - $f^0(x_1^0, \ldots, x_n^0) \neq f^1(x_1^1, \ldots, x_n^1)$ *OR*
    $f^0(x_1^0, \ldots, x_n^0) \neq f^1(x_1^0, \ldots, x_n^0)$ *OR*
    $f^1(x_1^0, \ldots, x_n^0) \neq f^1(x_1^1, \ldots, x_n^1)$.

  Phrased differently, we require for an adversary $\mathcal{A}$ to be admissible that

$$f^0(x_1^0, \ldots, x_n^0) = f^1(x_1^0, \ldots, x_n^0) = f^1(x_1^1, \ldots, x_n^1)$$

  for all $\mathsf{tag} \in \mathsf{Tag}$, vectors $(x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]} \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ and functions (f^0, f^1) specified by parameters $(y_i^0, y_i^1)_{i=1}^n$, where $x_i^0 = x_i^1$ and $y_i^0 = y_i^1$ for all $i \in \mathcal{C}$, and $(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ and $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ for all $i \in \mathcal{H}$. The experiment in this weak function-hiding model is denoted by $\mathbf{Exp}^{\mathsf{adap\text{-}dyn\text{-}wfh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

As usual, one can also consider experiments that implement a combination of the above restrictions. To emphasize the difference from weakly function-hiding, we sometimes refer to (standard) function-hiding as *fully* function-hiding.

Remark 5. The *complete queries* constraint 1 in the definition of non-admissible adversaries was first introduced in [24]. In the successive studies [25, 26] and in other results on the subject [6, 30, 2, 1], this condition can be removed. In Section 6.1, we first give a concrete construction satisfying the security notion as defined by Definition 4 having the foregoing constraint. Afterwards, in Section 6.3, we present a generic transformation to leverage the aforementioned construction so that it is secure even when condition 1 is not enforced.

Lemma 6 uses a standard hybrid reduction to prove that in the weakly function-hiding setting, single-challenge security is equivalent to multi-challenge security. The proof is given in Appendix B.1 for completeness.

Lemma 6. *Let* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a DMCFE scheme for the function class* $\mathcal{F}$. *If* $\mathcal{E}$ *is single-challenge weakly function-hiding (against static corruption), then it is also weakly function-hiding (against static corruption). More specifically, for any* ppt *adversary* $\mathcal{A}$, *there exists a* ppt *algorithm* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \leq \max(q_e, q_k) \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \ ,$$

where q_e *and* q_k *denote the maximum numbers of different tags* tag *and* $\mathsf{tag\text{-}f}$ *that* $\mathcal{A}$ *can query to* $\mathcal{O}\mathsf{Enc}$ *and* $\mathcal{O}\mathsf{DKeyGen}$ *respectively.*

The works of [42] and [4] present generic transformations that turn weakly function-hiding (multi-input) functional encryption schemes into full-fledged function-hiding schemes. A similar transformation, stated in Lemma 7, is also applicable in the case of FH-DMCFE. The proof is very similar to [42, 4] but given in Appendix B.2 for completeness.

Lemma 7. *Let $\mathcal{F}_n^{\mathsf{IP}}$ denote the function class containing inner products of length n and let $\mathcal{E}$ be a weakly function-hiding DMCFE scheme for $\mathcal{F}_{2n}^{\mathsf{IP}}$. Then there exists a (fully) function-hiding DMCFE scheme $\mathcal{E}'$ for $\mathcal{F}_n^{\mathsf{IP}}$. More precisely, for any* ppt *adversary $\mathcal{A}$, there exists a* ppt *algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}fh}}_{\mathcal{E}',\mathcal{F}_n^{\mathsf{IP}},\mathcal{A}}(1^\lambda) \leq 3 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}wfh}}_{\mathcal{E},\mathcal{F}_{2n}^{\mathsf{IP}},\mathcal{B}}(1^\lambda) \ ,$$

where $\mathsf{xxx} \in \{\mathsf{adap}, \mathsf{sel}\}$ *and* $\mathsf{yyy} \in \{\mathsf{dyn}, \mathsf{stat}\}$.

5 Secret-Sharing Swapping Lemma

In this section we state and prove a technical lemma that will be the basis of our results and may be of independent interest, with various applications.

Lemma 8 (Secret-Sharing Swapping). *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda) \in \mathbb{N}$ where $H, K, L : \mathbb{N} \to \mathbb{N}$ are functions. Let $(\mathbf{B}_i, \mathbf{B}_i^*)_{i\in[H]}$ be two random dual bases of dimension 9 in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathbf{t}, g_1, g_2, g_\mathbf{t}, \mathbf{e}, q)$. All basis vectors are kept secret.*
We consider any public values $(x_i, x_{\ell,i}, x'_{\ell,i}, y_i^0, y_i^1, y_{k,i}, R, R_k)_{i\in[H],k\in[K],\ell\in[L]}$ such that $\sum_{i=1}^H x_i y_i^0 = \sum_{i=1}^H x_i y_i^1$. With random $r, r_\ell, \rho_i, \rho_{\ell,i}, \pi_i, \pi_{k,i}, \sigma_i, \sigma_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ such that $\sum_{i=1}^H \sigma_i = R$ and $\sum_{i=1}^H \sigma_{k,i} = R_k$, for all $k \in [K]$, the following distributions are computationally indistinguishable under the SXDH *assumption:*

$$D_0 := \left\{ \begin{array}{l} (\mathbf{c}_{\ell,i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i})_{\ell\in[L]} \\ \left(\mathbf{c}_i = (\boxed{x_i},\ \boxed{0},\ r,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}\right)_{i\in[H]} \\ \left(\mathbf{d}_i = (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H]} \\ \left(\mathbf{d}_{k,i} = (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H],k\in[K]} \end{array} \right\};$$

$$D_1 := \left\{ \begin{array}{l} (\mathbf{c}_{\ell,i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i})_{\ell\in[L]} \\ \left(\mathbf{c}_i = (\boxed{0},\ \boxed{x_i},\ r,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}\right)_{i\in[H]} \\ \left(\mathbf{d}_i = (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H]} \\ \left(\mathbf{d}_{k,i} = (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H],k\in[K]} \end{array} \right\}.$$

More specifically, we have:

$$\left| \Pr_{\mathsf{samp}\sim D_0}[\mathcal{A}(\mathsf{samp}) \to 1] - \Pr_{\mathsf{samp}\sim D_1}[\mathcal{A}(\mathsf{samp}) \to 1] \right| \leq 12 \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

for any specific ppt *$\mathcal{A}$ with fixed $(R, (x_i, (x_{\ell,i}, x'_{\ell,i})_{\ell=1}^L, y_i^0, y_i^1, (y_{k,i}, R_k)_{k=1}^K)_{i=1}^H)$.*

The proof of Lemma 8 can be found in Appendix B.3. Below we give the main ideas of the demonstration.

Game G_0: The vectors are sampled according to D_0.
Game G_1: (Random 0-Secret Sharing)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& x_i & 0 & \boxed{r} & 0 & \rho_i & 0 & 0 & 0 & \boxed{r'} &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \boxed{\sigma_i} & \pi_i & 0 & 0 & 0 & 0 & \boxed{\tau_i} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_2: (Formal Duplication from 1 to 7 and from 2 to 8 in $\mathbf{B}_i^*$)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& x_i & 0 & r & 0 & \rho_i & 0 & 0 & 0 & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & 0 & \boxed{y_i^1} & \boxed{y_i^0} & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & \boxed{y_{k,i}} & \boxed{y_{k,i}} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_3: (Computational Swapping between 1 and 7 in $\mathbf{c}_i$ using 5-randomness in $\mathbf{B}_i$)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& \boxed{0} & 0 & r & 0 & \boxed{\rho_i} & 0 & \boxed{x_i} & 0 & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & 0 & y_i^1 & y_i^0 & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_4: (Formal Duplication from 7 to 6 in $\mathbf{B}_i$)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & \boxed{x_i} & x_i & 0 & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & 0 & y_i^1 & y_i^0 & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_5: $\Delta y_i \coloneqq y_i^1 - y_i^0$ (Formal Swapping)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & \boxed{0} & \boxed{0} & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & x_i & \boxed{0} & \boxed{x_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{\Delta y_i} & y_i^1 & y_i^0 & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & \boxed{0} & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_6: (Formal Quotient)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & \boxed{r'} & 0 & x_i & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{x_i \Delta y_i / r'} & y_i^1 & y_i^0 & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_7: $\tau'_i \coloneqq \tau_i + x_i \Delta y_i / r'$ (Formal Cancelling)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & \boxed{0} & 0 & x_i & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{x_i \Delta y_i / r'} & y_i^1 & y_i^0 & \boxed{\tau'_i} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_8: $\alpha_i \coloneqq -(x_i \Delta y_i)/(r' \tau'_i)$ (Formal Randomizing)

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & 0 & 0 & x_i & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{0} & y_i^1 & y_i^0 & \boxed{\tau'_i} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Game G_9: Undo G_3, G_2, G_1 (Cleaning) – Vectors sampled according to D_1.

$$\begin{array}{lccccccccccl}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & \boxed{x_i} & r & 0 & \boxed{\rho_i} & 0 & 0 & \boxed{0} & \boxed{0} &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{0} & \boxed{0} & \boxed{0} & \boxed{0} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & \boxed{0} & \boxed{0} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

Fig. 2: Games for proving Lemma 8

Proof (Main ideas). The sequence of games is given in Figure 2. We explain the main steps in our proof (see Appendix B.3) as follows. We start from the game where the sample given to the adversary $\mathcal{A}$ follows $\mathbf{D}_0$. Our first step is to exploit the fact that $r \xleftarrow{\$} \mathbb{Z}_q$ is a uniformly random value and all the secret shares σ_i in $\mathbf{d}_i$ sum to a known constant R, *i.e.* $\sum_{i=1}^{H} \sigma_i = R$. This helps us perform a computational basis change on $(\mathbf{B}_i, \mathbf{B}_i^*)$ and introduce a value $r' \xleftarrow{\$} \mathbb{Z}_q^*$ in $\mathbf{c}_i[9]$ and a random secret sharing of 0 namely $(\tau_i)_{i=1}^H$ in $(\mathbf{d}_i[9])_{i=1}^H$:

$$\begin{array}{lccccccccccc}
\mathbf{c}_i = (& x_i & 0 & \boxed{r} & 0 & \rho_i & 0 & 0 & 0 & \boxed{r'} &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \boxed{\sigma_i} & \pi_i & 0 & 0 & 0 & 0 & \boxed{\tau_i} &)_{\mathbf{B}_i^*}
\end{array}$$

This change goes indistinguishable under the SXDH assumption, thanks to the fact that all vectors are kept secret. Moreover, we can only introduce a random secret sharing $(\tau_i)_{i=1}^H$ of 0 because this computational change intuitively "extracts" a 0-secret sharing from $(\sigma_i)_{i=1}^H$ into the 9-th coordinates of $\mathbf{d}_i$, which is the only way to preserve the relation $\sum_{i=1}^{H} \sigma_i = R$ and keep the summation over $i \in [H]$ of products between $\mathbf{c}$-vectors and $\mathbf{d}$-vectors invariant.

Next, we prepare the swapping of x_i in $\mathbf{c}_i$: we duplicate (y_i^1, y_i^0) (respectively, $(y_{k,i}, y_{k,i})$) from coordinates $(1, 2)$ to coordinates $(7, 8)$ in $\mathbf{d}_i$ (respectively, in $\mathbf{d}_{k,i}$). Afterwards, we use *computational* basis changes to move x_i from coordinate 1 to coordinate 7 in $\mathbf{c}_i$. We emphasize that all the basis changes we do so far allow simulating $\mathbf{c}_{\ell,i}$ so that they are not effected. We are now in a game where the adversary receives the vectors of the form in Game G_3 of Figure 2. The goal of this *isolation* of x_i is to do the swapping in coordinates $(7, 8)$ and avoid having affects on coordinates $(1, 2)$, otherwise we risk modifying coordinates $(1, 2)$ of $\mathbf{c}_{\ell,i}, \mathbf{d}_{k,i}$ and change the global view of $\mathcal{A}$. An astute reader might wonder why we cannot use the same *computational* change as we have used to move x_i between coordinates $(1, 7)$ of $\mathbf{c}_i$. The main reason is that $\mathbf{d}_i[1] = \mathbf{d}_i[7] = y_i^1$ and $\mathbf{d}_{k,i}[1] = \mathbf{d}_{k,i}[7] = y_{k,i}$, whereas $\mathbf{d}_i[1] \neq \mathbf{d}_i[2]$ and for this latter case we do not know how a computational change for such a swap would look like if we do it in place.

Given that (x_i, y_i^0, y_i^1) are now isolated to coordinates $(7, 8)$, we use *formal* changes in our subsequent transitions to swap x_i from coordinate 7 to 8 in $\mathbf{c}_i$ for the sake of a perfect indistinguishability in the views of $\mathcal{A}$. The formal argument starts by a formal duplication of x_i from coordinate 7 to coordinate 6, which will affect *all* vectors in $\mathbf{B}_i$. The other $\mathbf{c}_{\ell,i}$ stays invariant because $\mathbf{c}_{\ell,i}[6] = 0$ thus the duplication changes nothing. Furthermore, the current formal duplication changes the *dual* vectors by adding their 6-th coordinates to their 7-th one; Fortunately, all $\mathbf{d}$-vectors have 0 at their 6-th slot and therefore are kept invariant. The adversary is now given the vectors of the form in Game G_4 of Figure 2. The succeeding step swaps x_i in coordinates $(7, 8)$ by performing another *formal* basis change to substract the copy of x_i in $\mathbf{c}_i[6]$ from $\mathbf{c}_i[7]$ and add the same copy to $\mathbf{c}_i[8]$. Again, this affects *all* vectors in $\mathbf{B}_i$ but because $\mathbf{c}_{\ell,i}[6] = \mathbf{c}_{\ell,i}[7] = \mathbf{c}_{\ell,i}[8] = 0$, we have $\mathbf{c}_{\ell,i}$ staying unchanged. In the dual basis, the difference between the 7-th and 8-th coordinates will move into the 6-th coordinate, for *all* $\mathbf{d}$-vectors. This introduces $\Delta y_i := y_i^1 - y_i^0$ in the 6-th coordinate of $\mathbf{d}_i$ and no modifications to $\mathbf{d}_{k,i}$ for $\mathbf{d}_{k,i}[7] = \mathbf{d}_{k,i}[8] = y_{k,i}$. The form of the vectors is now:

$$\begin{array}{lccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & \boxed{0} & \boxed{0} & 0 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & x_i & \boxed{0} & \boxed{x_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{\Delta y_i} & y_i^1 & y_i^0 & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & \boxed{0} & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

An ensuing problem is the cleaning of the 6-th coordinate of $\mathbf{c}_i, \mathbf{d}_i$. To clean $\mathbf{c}_i[6]$, we first need a quotient done by a formal basis change to make $\mathbf{c}_i[6] = \mathbf{c}_i[9] = r'$, but this leads to two problems: (1) r' must be non-zero, because in either of the dual pair of bases there will be a factor $1/r'$ and (2) the basis change depends on x_i that will be needed since the beginning of the game, before giving

the sample to $\mathcal{A}$. We solve (1) since by constraining $r' \xleftarrow{\$} \mathbb{Z}_q^*$ from the first computational changes. To deal with (2), we notice that we are *in the middle of a formal argument* thus we can guess x_i since the beginning and the loss by this guess does not amplify the difference of advantages, as it is 0 due to the perfectly indistinguishable nature. After making $\mathbf{c}_i[6] = \mathbf{c}_i[9] = r'$, a formal basis change can be done to make *all* $\mathbf{c}$-vectors having 0 at their 6-th coordinate, by substracting the 9-th from the 6-th (for $\mathbf{c}_{\ell,i}[6]$ this is trivial). However, in the dual basis it leads to

$$\begin{array}{lccccccccc} \mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{x_i \Delta y_i / r'} & y_i^1 & y_i^0 & \boxed{\tau_i'} &)_{\mathbf{B}_i^*} \\ \mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*} \end{array}$$

where $\tau_i' := \tau_i + x_i \Delta y_i / r'$. Our key observation that due to the constraint $\sum_{i=1}^H x_i y_i^0 = \sum_{i=1}^H x_i y_i^1$, the values $(\tau_i')_{i=1}^H := (\tau_i + x_i \Delta y_i / r')_{i=1}^H$ will stay a random secret sharing of 0. Therefore, if $\tau_i' \neq 0$ we can perform a formal basis change to clean $\mathbf{d}_i[6]$ by substracting some multiple of $\mathbf{d}_i[9] = \tau_i'$. This is feasible because, all over again, we are *in the middle of a formal argument*: we guess from the beginning $x_i \Delta y_i / r'$, sample the two 0-secret sharings $\tau_i, \tau_i' \neq 0$ and continue the simulation only if $\tau_i' - \tau_i = x_i \Delta y_i / r'$. This ensures that we will continue the simulation only if all $\tau_i' \neq 0$, then perform a formal basis change based on $\alpha_i := -(x_i \Delta y_i)/(r' \tau_i')$ to arrive at a game whose vectors for $\mathcal{A}$ are:

$$\begin{array}{lccccccccc} \mathbf{c}_{\ell,i} = (& x_{\ell,i} & x'_{\ell,i} & r_\ell & 0 & \rho_{\ell,i} & 0 & 0 & 0 & 0 &)_{\mathbf{B}_i} \\ \mathbf{c}_i = (& 0 & 0 & r & 0 & \rho_i & 0 & 0 & x_i & r' &)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& y_i^1 & y_i^0 & \sigma_i & \pi_i & 0 & \boxed{0} & y_i^1 & y_i^0 & \boxed{\tau_i'} &)_{\mathbf{B}_i^*} \\ \mathbf{d}_{k,i} = (& y_{k,i} & y_{k,i} & \sigma_{k,i} & \pi_{k,i} & 0 & 0 & y_{k,i} & y_{k,i} & 0 &)_{\mathbf{B}_i^*} \end{array}$$

Given that x_i is in $\mathbf{c}_i[8]$ and $\mathbf{d}_i[8] = \mathbf{d}_i[2] = y_i^0$, a computational swap would move back x_i to the 2-nd coordinate of $\mathbf{c}_i$ without any problem. At the end, we redo the duplication in $\mathbf{d}_i, \mathbf{d}_{k,i}$ as well as the removal of r' and the random 0-secret shares $(\tau_i')_i$, in order to make the vectors follow D_1. □

6 An Adaptively Secure FH-DMCFE for Inner Products

6.1 A Function-Hiding DMCFE Secure Against One Adaptive Challenge with Complete Queries and Static Corruption

This section presents a function-hiding DMCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the function class $\mathcal{F}^{\mathsf{IP}} = \{F_{\mathbf{y}}\}$ where $F_{\mathbf{y}} : \left(\mathbb{Z}_q^*\right)^n \to \mathbb{Z}_q$ is defined as $F_{\mathbf{y}}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle$. We work in the prime-order bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$ and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}$ are all written additively. We employ two full-domain hash functions $\mathsf{H}_1 : \mathsf{Tag} \to \mathbb{G}_1$ and $\mathsf{H}_2 : \mathsf{Tag} \to \mathbb{G}_2$.
The details of $\mathcal{E}$ go as follows:

$\mathsf{Setup}(1^\lambda)$**:** Choose n pairs of dual orthogonal bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ for $i \in [n]$, where $(\mathbf{B}_i, \mathbf{B}_i^*)$ is a pair of dual bases for $(\mathbb{G}_1^{14}, \mathbb{G}_2^{14})$. For each $i \in [n]$, we denote j-th row of $\mathbf{B}_i$ (resp. $\mathbf{B}_i^*$) by $\mathbf{b}_{i,j}$ (resp. $\mathbf{b}_{i,j}^*$). Generate two random n-out-of-n secret sharings of 0, namely $(\tilde{s}_i)_i, (\tilde{t}_i)_i \xleftarrow{\$} \mathbb{Z}_q^n$ such that $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$. Then, output the *secret keys* sk_i and the *encryption keys* ek_i as follows:

$$\begin{aligned} \mathsf{sk}_i &:= (\mathbf{b}_{i,1}^*,\ \tilde{s}_i B_{i,2}^* + B_{i,8}^*,\ \mathbf{b}_{i,9}^*) \\ \mathsf{ek}_i &:= (\mathbf{b}_{i,1},\ \mathbf{b}_{i,10},\ B_{i,2} + \tilde{t}_i B_{i,8}) \end{aligned}$$

where $B_{i,k}$ (respectively $B_{i,k}^*$) denotes the k-th row of the basis changing matrix B_i (respectively B_i^*) for $i \in [n]$.

$\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i)$**:** Parse $\mathsf{sk}_i = (\mathbf{b}^*_{i,1},\ \tilde{s}_i B^*_{i,2} + B^*_{i,12},\ \mathbf{b}^*_{i,8})$. Compute $\mathsf{H}_2(\mathsf{tag\text{-}f}) \to [\![\mu]\!]_2 \in \mathbb{G}_2$ and sample $\pi_i \xleftarrow{\$} \mathbb{Z}_q$. Then compute and output

$$\begin{aligned}\mathbf{d}_i &= y_i \mathbf{b}^*_{i,1} + (\tilde{s}_i B^*_{i,2} + B^*_{i,8}) \cdot [\![\mu]\!]_2 + \pi_i \mathbf{b}^*_{i,9} \\ &= (y_i,\ \tilde{s}_i\mu,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i}\ .\end{aligned}$$

$\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i)$**:** Parse $\mathsf{ek}_i = (\mathbf{b}_{i,1},\ \mathbf{b}_{i,10},\ B_{i,2} + \tilde{t}_i B_{i,8})$. Compute $\mathsf{H}(\mathsf{tag}) \to [\![\omega]\!]_1 \in \mathbb{G}_1$ and sample a random scalar $\rho_i \xleftarrow{\$} \mathbb{Z}_q$. Finally, compute and output

$$\begin{aligned}\mathbf{c}_i &= x_i \mathbf{b}_{i,1} + \rho_i \mathbf{b}_{i,10} + (B_{i,2} + \tilde{t}_i B_{i,8}) \cdot [\![\omega]\!]_1 \\ &= (x_i,\ \omega,\ 0,\ 0,\ 0,\ 0,\ 0,\ \tilde{t}_i\omega,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}\ .\end{aligned}$$

$\mathsf{Dec}(\mathbf{d}, \mathbf{c})$**:** Parse $\mathbf{d} := (\mathbf{d}_i)_{i\in[n]}$ and $\mathbf{c} := (\mathbf{c}_i)_i$. Compute $[\![\mathsf{out}]\!]_\mathsf{t} = \prod_{i=1}^n \mathbf{c}_i \times \mathbf{d}_i$, then find and output the discrete log out.

Correctness. The correctness property is demonstrated as follows:

$$\begin{aligned}[\![\mathsf{out}]\!]_\mathsf{t} = \prod_{i=1}^n \mathbf{c}_i \times \mathbf{d}_i &= \prod_{i=1}^n [\![x_i y_i + \tilde{s}_i\mu\omega + \tilde{t}_i\omega\mu]\!]_\mathsf{t} \\ &= \left[\!\!\left[\langle \mathbf{x}, \mathbf{y}\rangle + \omega\mu \cdot \sum_{i=1}^n (\tilde{s}_i + \tilde{t}_i) \right]\!\!\right]_\mathsf{t} \\ &= [\![\langle \mathbf{x}, \mathbf{y}\rangle]\!]_\mathsf{t}\ ,\end{aligned}$$

and we are using the fact that $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$.

Security. Theorem 9 states that the scheme is *function-hiding* under *static corruption* and a *single adaptive challenge*. As discussed in Remark 5, we prove the security while employing the constraint that if an adversary queries for an honest component, either for some ciphertext (or for some key), then all honest components of the same ciphertext (or key) must also be queried. In Section 6.3, we show how to circumvent this constraint.

Theorem 9. *The DMCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* adaptively one-challenge function-hiding *in the ROM, under* static *corruption, if the* SXDH *assumption holds for* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*. More specifically, let* q_e *denote the number of challenge queries for identical messages, and* q_k *denote the number of functional key queries. Then, for any* ppt *adversary* $\mathcal{A}$ *against* $\mathcal{E}$*, we have the following bound:*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \leq (26q_e + 14q_k + 32) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

Proof (Main ideas). In Figure 3, we present the sequence of games used to prove Theorem 9, where we start from G_0 corresponding to $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}fh\text{-}0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, until arriving at G_8 which equals $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}fh\text{-}1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$. The changes that make the transition between games are highlighted by a frame. The full proof can be found in Appendix B.4.

The general plan for the proof can be revisited in the technical overview of Section 3.2, where we describe the purpose of the *pre-processing* ($\mathsf{G}_2 \to \mathsf{G}_3$), the main *complexity leveraging* step ($\mathsf{G}_3 \to \mathsf{G}_4$), and the *post-processing* ($\mathsf{G}_4 \to \mathsf{G}_6$). Whereas the pre-processing and the complexity leveraging is rather straightforward, the subsequent post-processing procedure, where we essentially undo the changes of the pre-processing, is technically much more involved. Now equipped with the necessary tool (Lemma 8), we can describe it in more detail. We recall that in G_4 (*i.e.* after the switch from (x^0_i, y^0_i) to (x^1_i, y^1_i) and before the pre-processing), we are roughly in the following state (coordinates used for purely technical aspects are omitted):

Game G_0: $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$, $\mathsf{H}(\mathsf{tag}_\ell) \to [\![\omega_\ell]\!]_1$, $\mathsf{H}_1(\mathsf{tag}^*) \to [\![\omega]\!]_1$, $\mathsf{H}_2(\mathsf{tag\text{-}f}_k) \to [\![\mu_k]\!]_2$, $\mathsf{H}_2(\mathsf{tag\text{-}f}^*) \to [\![\mu]\!]_2$

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & 0 & \tilde{t}_i\omega_\ell & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & \tilde{s}_i\mu_k & 0 & 0 & 0 & 0 & 0 & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & 0 & 0 & 0 & 0 & \tilde{t}_i\omega & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^0 & \tilde{s}_i\mu & 0 & 0 & 0 & 0 & 0 & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_1: $\sum_{i=1}^n s_{k,i} = \sum_{i=1}^n s_i = \sum_{i=1}^n t_{\ell,i} = \sum_{i=1}^n t_i = 0$ (Randomization)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & 0 & \boxed{t_{\ell,i}} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & \boxed{s_{k,i}} & 0 & 0 & 0 & 0 & 0 & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & 0 & 0 & 0 & 0 & \boxed{t_i} & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^0 & \boxed{s_i} & 0 & 0 & 0 & 0 & 0 & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_2: (Random 0-Secret Sharings)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & \boxed{r_\ell} & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & 0 & 0 & 0 & 0 & \boxed{\sigma_{k,i}} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & 0 & 0 & 0 & \boxed{r} & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^0 & s_i & 0 & 0 & 0 & 0 & \boxed{\sigma_i} & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_3: (Pre-Processing, see Figure 5 for details)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& \boxed{0} & \boxed{0} & \boxed{x_{\ell,i}} & 0 & \boxed{\omega_\ell} & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& \boxed{0} & \boxed{0} & \boxed{y_{k,i}} & \boxed{y_{k,i}} & \boxed{s_{k,i}} & \boxed{s_{k,i}} & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & \boxed{x_i^0} & 0 & \boxed{\omega} & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^0 & s_i & \boxed{y_i^0} & 0 & \boxed{s_i} & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_4: $s_i' \coloneqq s_i - (x_i^1 y_i^1 - x_i^0 y_i^0)/\omega$ (Complexity Leveraging)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_i = (& \boxed{x_i^1} & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& \boxed{y_i^1} & \boxed{s_i'} & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_5: (Swapping)

$$
\mathbf{d}_{k,i} = (\quad 0 \quad \boxed{s_{k,i}} \quad y_{k,i} \quad y_{k,i} \quad s_{k,i} \quad \boxed{0} \quad \sigma_{k,i} \quad \mu_k \quad \pi_{k,i} \quad 0 \quad 0^4 \quad)_{\mathbf{B}_i^*}
$$

Game G_6: (Secret-Sharing Swapping (Lemma 8), see Figure 6 for details)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& \boxed{x_{\ell,i}} & \boxed{\omega_\ell} & \boxed{0} & 0 & \boxed{0} & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& \boxed{y_{k,i}} & s_{k,i} & y_{k,i} & \boxed{0} & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_7: (Cleaning, see Figure 7 for details)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & \boxed{0} & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & \boxed{0} & 0 & \boxed{0} & 0 & \boxed{0} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & \boxed{0} & 0 & \boxed{0} & \boxed{0} & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s_i' & \boxed{0} & 0 & \boxed{0} & 0 & \boxed{0} & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_8: $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$ (De-Randomization)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & 0 & \boxed{\tilde{t}_i\omega_\ell} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & \boxed{\tilde{s}_i\mu_k} & 0 & 0 & 0 & 0 & 0 & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & 0 & 0 & 0 & 0 & \boxed{\tilde{t}_i\omega} & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & \boxed{\tilde{s}_i\mu} & 0 & 0 & 0 & 0 & 0 & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Fig. 3: Games for proving Theorem 9

$$\begin{array}{rccc|c|c|c|c|l} \mathbf{c}_{\ell,i} = (& 0 & 0 & x_{\ell,i} & 0 & \omega_\ell & 0 & t_{\ell,i} &)_{\mathbf{B}_i} \\ \mathbf{d}_{k,i} = (& 0 & 0 & y_{k,i} & y_{k,i} & s_{k,i} & s_{k,i} & \mu_k &)_{\mathbf{B}_i^*} \\ \mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & t_i &)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& y_i^1 & s_i' & y_i^0 & 0 & s_i & 0 & \mu &)_{\mathbf{B}_i^*} \end{array}$$

Our aim is to move $(x_{\ell,i}, y_{k,i})$ as well as $(\omega_\ell, s_{k,i})$ back to positions $(1,2)$ of $(\mathbf{c}_{\ell,i}, \mathbf{d}_{k,i})$ to be correctly in the game whose challenge bit is $b = 1$. First, we note that it is easy to move $s_{k,i}$ to coordinate 2, as all ciphertexts $\mathbf{c}_i$ and $\mathbf{c}_{\ell,i}$ have equal entries in their facing coordinates $(2,6)$: $\mathbf{c}_i[2] = \mathbf{c}_i[6] = \omega$ and $\mathbf{c}_{\ell,i}[2] = \mathbf{c}_{\ell,i}[6] = 0$. Therefore, the entries $(2,6)$ of $\mathbf{d}_{k,i}$ can be swapped under DDH in $\mathbb{G}_2$ which is applied in G_5. Unfortunately, the story is not that easy for the remaining three cases $x_{\ell,i}$, $y_{k,i}$ and ω_ℓ. For example, when moving $x_{\ell,i}$ back to coordinate 1, we change the *local* inner product $\langle \mathbf{c}_{\ell,i}, \mathbf{d}_i \rangle$ since in general $\mathbf{d}_i[3] = y_i^0 \neq y_i^1 = \mathbf{d}_i[1]$. Nevertheless, under the *function-hiding* property, the *global* inner product over all $i \in [n]$ is invariant for ℓ:

$$\sum_{i=1}^{n} x_{\ell,i} y_i^0 = \sum_{i=1}^{n} x_{\ell,i} y_i^1$$

In particular, the admissibility condition of Definition 4 gives:

$$\sum_{\text{honest } i} x_{\ell,i} y_i^0 = \sum_{\text{honest } i} x_{\ell,i} y_i^1 \ . \tag{2}$$

due to the constraint that $x_i^0 = x_i^1$ and $y_i^0 = y_i^1$ for all *corrupted* i. Thus, there is some hope that the local inner products may be randomized in a way that an adversary cannot efficiently detect local changes while preserving the global inner product. It turns out that this hope is justified, though by a complicated proof. The centerpiece is the secret-sharing swapping (Lemma 8) from Section 5. Before going into the details of how this lemma can be applied, we mention that similar to (2) for $x_{\ell,i}$, there are connections that help to move back $y_{k,i}$ and ω_ℓ. Specifically, we have (even under standard security without the function-hiding property):

$$\sum_{i=1}^{n} x_i^0 y_{k,i} = \sum_{i=1}^{n} x_i^1 y_{k,i} \ ,$$

and in particular:

$$\sum_{\text{honest } i} x_i^0 y_{k,i} = \sum_{\text{honest } i} x_i^1 y_{k,i} \ . \tag{3}$$

Additionally, we also make use of the equation

$$\sum_{\text{honest } i} \omega_\ell s_i = \sum_{\text{honest } i} \omega_\ell s_i' \ , \tag{4}$$

where $s_i' = \mathbf{d}_i[2]$ since G_4, while noting that $s_i' = s_i$ for all $i \in \mathcal{C}$ because of the admissibility of *function-hiding* setting.

From G_5 to G_6, we need to apply the secret sharing swapping in a specific order.

We first remark that because we are in the *static* corruption setting, the secret shares $(s_i, s_{k,i})_{i \in \mathcal{H}}$ and $(t_i, t_{\ell,i})_{i \in \mathcal{H}}$ sum to fixed values, *i.e.*

$$\sum_{i \in \mathcal{H}} s_{k,i} = \mu_k \sum_{i \in \mathcal{H}} \tilde{s}_i, \ \sum_{i \in \mathcal{H}} s_i = \mu \sum_{i \in \mathcal{H}} \tilde{s}_i \quad \text{and} \quad \sum_{i \in \mathcal{H}} t_{\ell,i} = \omega_\ell \sum_{i \in \mathcal{H}} \tilde{t}_i, \ \sum_{i \in \mathcal{H}} t_i = \omega \sum_{i \in \mathcal{H}} \tilde{t}_i$$

while $\sum_{i\in\mathcal{H}} \tilde{s}_i$ and $\sum_{i\in\mathcal{H}} \tilde{t}_i$ are known from setup, the two common constant $\mu, \omega, \mu_k, \omega_\ell \xleftarrow{\$} \mathbb{Z}_q$ are given by the RO. We remark that for applying the secret-sharing swapping lemma, we also change the random secret sharings in the $\mathbf{d}$-vectors and in the $\mathbf{c}$-vectors from shiftings $(\mu\tilde{s}_i, \mu_k\tilde{s}_i)_i$ and $(\omega\tilde{t}_i, \omega_\ell\tilde{t}_i)_i$ of secret sharing of 0, where $(\tilde{s}_i)_i, (\tilde{t}_i)_i$ are generated at setup, to random independent secret sharings $(s_i, s_{k,i})_i$ and $(t_i, t_{\ell,i})_i$, respectively. This can be done using DDH and its *random self-reducibility*, while noting that we can embed the sum of the honest shares, which is fixed due to *static* corruption, into the exponents of group elements to correctly simulate the honest key components. The embedding works by first generating all except one (randomly self-reduced) honest shares, then use linearity on the exponent to define the last one. The simulation works because for honest i, we never have to compute in the clear the exponents.
Now, given all the ingredients for the lemma, the swappings are done specifically as follows:

1. First, we switch ω_ℓ from coordinate 6 back to coordinate 2 of $\mathbf{c}_{\ell,i}$, while $\mathbf{d}_i[2] = s'_i$,$\mathbf{d}_i[6] = s_i$, and $\mathbf{d}_{k,i}[2] = \mathbf{d}_{k,i}[6] = s_{k,i}$. We employ condition (4), use $r, r_\ell \xleftarrow{\$} \mathbb{Z}_q$ (introduced in G_2) in $\mathbf{c}_i[7], \mathbf{c}_{\ell,i}[7]$ as the fixed randomness, together with the secret shares are $\sigma_i, \sigma_{k,i}$ in $\mathbf{d}_i[7], \mathbf{d}_{k,i}[7]$ summing to a fixed constant over all $i \in \mathcal{H}$, which we also introduced from previous games.
2. Secondly, we switch $y_{k,i}$ from coordinate 4 back to coordinate 1 of $\mathbf{d}_{k,i}$, while $\mathbf{c}_i[1] = x_i^1$,$\mathbf{c}_i[4] = x_i^0$, and $\mathbf{c}_{\ell,i}[1] = \mathbf{c}_{\ell,i}[4] = x_{\ell,i}$. We employ condition (3), use $\mu, \mu_k \xleftarrow{\$} \mathbb{Z}_q$ (by the RO) in $\mathbf{d}_i[8], \mathbf{d}_{k,i}[8]$ as the fixed randomness (namely "r, r_ℓ" in the lemma's statement, we are in the dual basis in this swapping), together with the secret shares are $t_{\ell,i} = \mathbf{c}_{\ell,i}[8], t_i = \mathbf{c}_i[8]$ summing to a fixed constant over all $i \in \mathcal{H}$, which we know up front thanks to static corruption.
3. Finally, we switch $x_{\ell,i}$ from coordinate 3 back to coordinate 1 of $\mathbf{c}_{\ell,i}$, while $\mathbf{d}_i[1] = y_i^1$,$\mathbf{d}_i[3] = y_i^0$, and $\mathbf{d}_{k,i}[1] = \mathbf{d}_{k,i}[3] = y_{k,i}$. We employ condition (2), use $r, r_\ell \xleftarrow{\$} \mathbb{Z}_q$ (introduced from previous games) in $\mathbf{c}_i[7], \mathbf{c}_{\ell,i}[7]$ as the fixed randomness, together with the secret shares are $\sigma_i, \sigma_{k,i}$ in $\mathbf{d}_i[7], \mathbf{d}_{k,i}[7]$ summing to a fixed constant over all $i \in \mathcal{H}$, which we also introduced from previous games.

After this three applications of Lemma 8, it remains cleaning (Game G_7) and un-programming the products $(\mu_k\tilde{s}_i, \mu\tilde{s}_i, \omega_\ell\tilde{t}_i, \omega\tilde{t}_i)$ using DDH (Game G_8). The view of the adversary in G_8 corresponds to $b = 1$ in the security experiment. □

6.2 An FH-DMCFE Secure Against Multiple Adaptive Challenges with Complete Queries and Static Corruption

In this section, we lift the scheme from Section 6.1 to a function-hiding DMCFE secure against *multiple challenges*. We denote the function class containing inner products of length n by $\mathcal{F}_n^{\mathsf{IP}}$. Let $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ denote the DMCFE scheme from Section 6.1 for the function class $\mathcal{F}_{2n}^{\mathsf{IP}}$. From an application of Theorem 9, it follows that $\mathcal{E}$ is *function-hiding* against *one challenge*. Lemma 6 implies that the same scheme is *weakly* function-hiding even against *multiple challenges*. We emphasize that whether a scheme is weakly or fully function-hiding in the single-challenge setting, an application of Lemma 6 always yields *weakly* function-hiding security in the multi-challenge case. Therefore, we have to apply the generic transformation of Lemma 7 to $\mathcal{E}$ afterwards. This yields a new scheme $\mathcal{E}' = (\mathsf{Setup}', \mathsf{DKeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ which preserves the multi-challenge security and additionally satisfies the *fully* function-hiding property. We describe the resulting scheme $\mathcal{E}'$:

$\mathsf{Setup}'(1^\lambda)$**:** Choose $2n$ pairs of dual orthogonal bases $(\mathbf{B}_i, \mathbf{B}_i^*)_{i\in[2n]}$ where $(\mathbf{B}_i, \mathbf{B}_i^*)$ is a pair of dual bases for $(\mathbb{G}_1^{14}, \mathbb{G}_2^{14})$. Sample two random $2n$-out-of-$2n$ secret sharings of 0 denoted $(\tilde{s}_i)_i, (\tilde{t}_i)_i \in$

$\mathbb{Z}_q^{2n}$ and output the *secret keys* sk_i and the *encryption keys* ek_i for $i \in [n]$ as follows:

$$\mathsf{sk}_i := (\mathbf{b}^*_{i,1},\ \tilde{s}_i B^*_{i,2} + B^*_{i,8},\ \mathbf{b}^*_{i,9},\ \tilde{s}_{n+i} B^*_{n+i,2} + B^*_{n+i,8},\ \mathbf{b}^*_{n+i,9})$$
$$\mathsf{ek}_i := (\mathbf{b}_{i,1},\ \mathbf{b}_{i,10},\ B_{i,2} + \tilde{t}_i B_{i,8},\ \mathbf{b}_{n+i,10},\ B_{n+i,2} + \tilde{t}_{n+i} B_{n+i,8})$$

$\mathsf{DKeyGen}'(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i)$**:** Parse $\mathsf{sk}_i = (\mathbf{b}^*_{i,1},\ \tilde{s}_i B^*_{i,2} + B^*_{i,8},\ \mathbf{b}^*_{i,9},\ \tilde{s}_{n+i} B^*_{n+i,2} + B^*_{n+i,8},\ \mathbf{b}^*_{n+i,9})$. Compute $\mathsf{H}_2(\mathsf{tag\text{-}f}) \to [\![\mu]\!]_2 \in \mathbb{G}_2$ and sample $\pi_i, \pi_{n+i} \xleftarrow{\$} \mathbb{Z}_q$. Then compute

$$\begin{aligned}
\mathbf{d}_i &= y_i \mathbf{b}^*_{i,1} + (\tilde{s}_i B^*_{i,2} + B^*_{i,8}) \cdot [\![\mu]\!]_2 + \pi_i \mathbf{b}^*_{i,9} \\
&= (y_i,\ \tilde{s}_i \mu,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i} \\
\mathbf{d}_{n+i} &= (\tilde{s}_{n+i} B^*_{n+i,2} + B^*_{n+i,8}) \cdot [\![\mu]\!]_2 + \pi_{n+i} \mathbf{b}^*_{n+i,9} \\
&= (0,\ \tilde{s}_{n+i} \mu,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu,\ \pi_{n+i},\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_{n+i}}
\end{aligned}$$

and output $(\mathbf{d}_i, \mathbf{d}_{n+i})$.

$\mathsf{Enc}'(\mathsf{ek}_i, \mathsf{tag}, x_i)$**:** Parse $\mathsf{ek}_i = (\mathbf{b}_{i,1},\ \mathbf{b}_{i,10},\ B_{i,2} + \tilde{t}_i B_{i,8},\ \mathbf{b}_{n+i,10},\ B_{n+i,2} + \tilde{t}_{n+i} B_{n+i,8})$. Compute $\mathsf{H}(\mathsf{tag}) \to [\![\omega]\!]_1 \in \mathbb{G}_1$ and sample two random scalars $\rho_i, \rho_{n+i} \xleftarrow{\$} \mathbb{Z}_q$. Finally, compute

$$\begin{aligned}
\mathbf{c}_i &= x_i \mathbf{b}_{i,1} + \rho_i \mathbf{b}_{i,10} + (B_{i,2} + \tilde{t}_i B_{i,8}) \cdot [\![\omega]\!]_1 \\
&= (x_i,\ \omega,\ 0,\ 0,\ 0,\ 0,\ 0,\ \tilde{t}_i \omega,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i} \\
\mathbf{c}_{n+i} &= \rho_{n+i} \mathbf{b}_{n+i,10} + (B_{n+i,2} + \tilde{t}_{n+i} B_{n+i,8}) \cdot [\![\omega]\!]_1 \\
&= (0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ 0,\ \tilde{t}_{n+i} \omega,\ 0,\ \rho_{n+i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_{n+i}}
\end{aligned}$$

and output $(\mathbf{c}_i, \mathbf{c}_{n+i})$.

$\mathsf{Dec}'(\mathbf{d}, \mathbf{c})$**:** Parse $\mathbf{d} := (\mathbf{d}_i, \mathbf{d}_{n+i})_{i \in [n]}$ and $\mathbf{c} := (\mathbf{c}_i, \mathbf{c}_{n+i})_{i \in [n]}$. Compute $[\![\mathsf{out}]\!]_{\mathbf{t}} = \prod_{i=1}^{2n} \mathbf{c}_i \times \mathbf{d}_i$, then find and output the discrete log out.

6.3 Adaptive Security against Static Corruption with Incomplete Queries

In this section, we present a generic transformation to make our DMCFE schems from Section 6.1 secure in a function-hiding setting where the adversary can omit to query some *honest* components, either in the challenge ciphertext or in the challenge functional key. We first give the formal definition of this stronger security model in Appendix A.1, adapted from Definition 4 by removing the constraint 1 on *complete* queries for admissible adversaries.

The transformation. We adapt the transformation in [1, Section 4]. Their transformation turns any DMCFE scheme (without function-hiding property), which is secure under the restriction that all honest *challenge ciphertext* components are queried, into a DMCFE scheme that is secure even when the adversary can omit some honest client in the challenge ciphertext queries. The transformation from [1, Section 4] makes use of symmetric-key encryption schemes as well as pseudorandom functions, but it deals only with incomplete challenge ciphertexts *as per* [1, Definition 2.7]. Naturally, we can extend the ideas of [1] to cover additionally the incomplete *challenge functional keys* in our *function-hiding* setting for DMCFE. We refer to the technical overview **Achieving security against incomplete queries.** in Section 3.2 for the main ideas.

We remark that our transformation incurs an increase of ciphertext's size to $\Theta(n)$ due to the fact that there are n additional PRF evaluations in ct'_i, and therefore the total cost of communication is $\Theta(n^2)$. In the work by Chotard *et al.* [26], the authors introduced the notion of *All-or-Nothing Encapsulation* (AoNE) that encapsulates data in way only when all encapsulated components

are gathered can we decapsulate and recover the orginal pieces of data. In [26], AoNE can be constructed in the bilinear group setting and is suitable for our constructions from Section 6.1. More importantly, in their IP-DDFE construction of [26, Section 7.2], Chotard *et al.* used AoNE in a generic way in order to deal with *incomplete queries*, while keeping the total communication $O(n)$ at the cost of *selective* (ciphertext) challenges under *static* corruption. We can apply the same generic transformation using AoNE as in [26, Section 7.2] to deal with incomplete queries, in the *selective multiple* (ciphertext and key) challenges under *static* corruption.
We use a symmetric-key scheme $\mathsf{SE} = (\mathsf{Enc}_{\mathsf{SE}}, \mathsf{Dec}_{\mathsf{SE}})$ and a pseudorandom function PRF. We recall the syntax and security notions for pseudorandom functions in Appendix A.2 and for symmetric-key encryption schemes in Appendix A.3. We refer to the technical overview in Section 3.2 and provide a proof of security in Appendix B.5. Our transformation preserves the weakly function-hiding property (implied by the fully FH) in the one adaptive challenge keys and ciphertexts, under static corruption (revisit Section 3.2 for the intuition), that is statisfied from Section 6.1. Suppose we have a DMCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$. We transform $\mathcal{E}$ into $\mathcal{E}' = (\mathsf{Setup}', \mathsf{DKeyGen}', \mathsf{Enc}', \mathsf{Dec}')$ as follows:

$\mathsf{Setup}'(1^\lambda)$**:** Run $\mathsf{Setup}(1^\lambda)$ to obtain the keys $(\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}$. Then, sample $2n^2$ PRF keys, specifically $\mathsf{k}^{\mathsf{ct}}_{i,j}, \mathsf{k}^{\mathsf{dk}}_{i,j} \xleftarrow{\$} \{0,1\}^\lambda$ for $i, j \in [n]$. Finally define

$$\mathsf{sk}'_i = (\mathsf{sk}_i, \{\mathsf{k}^{\mathsf{dk}}_{i,j}, \mathsf{k}^{\mathsf{dk}}_{j,i}\}_{j\in[n]}); \quad \mathsf{ek}'_i = (\mathsf{ek}_i, \{\mathsf{k}^{\mathsf{ct}}_{i,j}, \mathsf{k}^{\mathsf{ct}}_{j,i}\}_{j\in[n]})$$

$\mathsf{DKeyGen}'(\mathsf{sk}'_i, \mathsf{tag\text{-}f}, y_i)$**:** Parse $\mathsf{sk}'_i = (\mathsf{sk}_i, \{\mathsf{k}^{\mathsf{dk}}_{i,j}, \mathsf{k}^{\mathsf{dk}}_{j,i}\}_{j\in[n]})$.
Compute $\mathsf{dk}_i \leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i)$. Then, compute $\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{j,i}, \mathsf{tag\text{-}f})$ for $j \in [n]$, then, $K^{\mathsf{dk}}_i(\mathsf{tag\text{-}f}) := \oplus_{j\in[n]}\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{i,j}, \mathsf{tag\text{-}f})$, and $\mathsf{dk}'_i := \mathsf{Enc}_{\mathsf{SE}}(K^{\mathsf{dk}}_i(\mathsf{tag\text{-}f}), \mathsf{dk}_i)$. Output $(\mathsf{dk}'_i, \{\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{j,i}, \mathsf{tag\text{-}f})\}_{j\in[n]})$.
$\mathsf{Enc}'(\mathsf{ek}'_i, \mathsf{tag}, x_i)$**:** Parse $\mathsf{ek}'_i = (\mathsf{ek}_i, \{\mathsf{k}^{\mathsf{ct}}_{i,j}, \mathsf{k}^{\mathsf{ct}}_{j,i}\}_{j\in[n]})$. Compute $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i)$. Then, compute $\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{j,i}, \mathsf{tag})$ for $j \in [n]$, $K^{\mathsf{ct}}_i(\mathsf{tag}) := \oplus_{j\in[n]}\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{i,j}, \mathsf{tag})$, and $\mathsf{ct}'_i := \mathsf{Enc}_{\mathsf{SE}}(K_i(\mathsf{tag}), \mathsf{ct}_i)$. Output $(\mathsf{ct}'_i, \{\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{j,i}, \mathsf{tag})\}_{j\in[n]})$.
$\mathsf{Dec}'(\mathbf{d}, \mathbf{c})$**:** Parse

$$\mathbf{d} = (\mathsf{dk}'_i, \{\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{j,i}, \mathsf{tag\text{-}f})\}_{j\in[n]})_{i\in[n]}, \quad \mathbf{c} = (\mathsf{ct}'_i, \{\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{j,i}, \mathsf{tag})\}_{j\in[n]})_{i\in[n]} \ .$$

Then, for each $j \in [n]$, compute

$$K^{\mathsf{ct}}_j(\mathsf{tag}) = \oplus_{i\in[n]}\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{j,i}, \mathsf{tag}); \qquad \mathsf{ct}_j = \mathsf{Dec}_{\mathsf{SE}}(K^{\mathsf{ct}}_j(\mathsf{tag}), \mathsf{ct}'_j)$$
$$K^{\mathsf{dk}}_j(\mathsf{tag\text{-}f}) = \oplus_{i\in[n]}\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{j,i}, \mathsf{tag\text{-}f}); \qquad \mathsf{dk}_j = \mathsf{Dec}_{\mathsf{SE}}(K^{\mathsf{dk}}_j(\mathsf{tag\text{-}f}), \mathsf{dk}'_j) \ .$$

Finally, compute and output $\mathsf{Dec}((\mathsf{dk}_i)_{i\in[n]}, (\mathsf{ct}_i)_{i\in[n]})$.

Acknowledgments

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

References

1. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *ASIACRYPT 2019, Part III*, 2019.
2. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC 2019, Part II*, 2019.
3. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC 2015*, 2015.
4. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO 2018, Part I*, 2018.
5. M. Abdalla, D. Catalano, R. Gay, and B. Ursu. Inner-product functional encryption with fine-grained access control. In *ASIACRYPT 2020, Part III*, 2020.

6. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT 2017, Part I*, 2017.
7. S. Agrawal, M. Clear, O. Frieder, S. Garg, A. O'Neill, and J. Thaler. Ad hoc multi-input functional encryption. In *ITCS 2020*, 2020.
8. S. Agrawal, R. Goyal, and J. Tomida. Multi-input quadratic functional encryption from pairings. In *CRYPTO 2021, Part IV*, 2021.
9. S. Agrawal, R. Goyal, and J. Tomida. Multi-party functional encryption. In *TCC 2021, Part II*, 2021.
10. S. Agrawal, R. Goyal, and J. Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In *TCC '22*. Springer-Verlag, 2022. `https://ia.cr/2022/1168`.
11. S. Agrawal, B. Libert, and D. Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *CRYPTO 2016, Part III*, 2016.
12. P. Ananth and A. Jain. Indistinguishability obfuscation from compact functional encryption. In *CRYPTO 2015, Part I*, 2015.
13. P. Ananth and A. Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In *EUROCRYPT 2017, Part I*, 2017.
14. N. Attrapadung, B. Libert, and E. de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *PKC 2011*, 2011.
15. C. E. Z. Baltico, D. Catalano, D. Fiore, and R. Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In *CRYPTO 2017, Part I*, 2017.
16. F. Benhamouda, F. Bourse, and H. Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In *PKC 2017, Part II*, 2017.
17. A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. In *ASIACRYPT 2015, Part I*, 2015.
18. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *56th FOCS*, 2015.
19. D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO 2001*, 2001.
20. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC 2011*, 2011.
21. G. Castagnos, F. Laguillaumie, and I. Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In *ASIACRYPT 2018, Part II*, 2018.
22. M. Chase. Multi-authority attribute based encryption. In *TCC 2007*, 2007.
23. J. Chen, H. W. Lim, S. Ling, H. Wang, and H. Wee. Shorter IBE and signatures via asymmetric pairings. In *PAIRING 2012*, 2013.
24. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT 2018, Part II*, 2018.
25. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. `https://eprint.iacr.org/2018/1021`.
26. J. Chotard, E. Dufour-Sans, R. Gay, D. H. Phan, and D. Pointcheval. Dynamic decentralized functional encryption. In *CRYPTO 2020, Part I*, 2020.
27. C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, 2001.
28. P. Datta, R. Dutta, and S. Mukhopadhyay. Functional encryption for inner product with full function privacy. In *PKC 2016, Part I*, 2016.
29. P. Datta, R. Dutta, and S. Mukhopadhyay. Strongly full-hiding inner product encryption. *Theoretical Computer Science*, 2017.
30. P. Datta, T. Okamoto, and J. Tomida. Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In *PKC 2018, Part II*, 2018.
31. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. Villar. An algebraic framework for Diffie-Hellman assumptions. In *CRYPTO 2013, Part II*, 2013.
32. R. Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In *PKC 2020, Part I*, 2020.
33. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F.-H. Liu, A. Sahai, E. Shi, and H.-S. Zhou. Multi-input functional encryption. In *EUROCRYPT 2014*, 2014.
34. S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from LWE. In *CRYPTO 2015, Part II*, 2015.
35. S. D. Gordon, J. Katz, F.-H. Liu, E. Shi, and H.-S. Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. `https://eprint.iacr.org/2013/774`.
36. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, 2006. Available as Cryptology ePrint Archive Report 2006/309.

37. S. Kim, J. Kim, and J. H. Seo. A new approach for practical function-private inner product encryption. Cryptology ePrint Archive, Report 2017/004, 2017. `https://eprint.iacr.org/2017/004`.
38. S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu. Function-hiding inner product encryption is practical. In *SCN 18*, 2018.
39. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC 2010*, 2010.
40. B. Libert and R. Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT 2019, Part III*, 2019.
41. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In *CRYPTO 2017, Part I*, 2017.
42. H. Lin and V. Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In *57th FOCS*, 2016.
43. K. Nguyen, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with fine-grained access-control. In *Asiacrypt '22*. Springer-Verlag, 2022. `https://ia.cr/2022/215`.
44. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO 2010*, 2010.
45. T. Okamoto and K. Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT 2012*, 2012.
46. T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT 2012*, 2012.
47. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In *ACM CCS 2007*, 2007.
48. A. Sahai and B. R. Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, 2005.
49. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO'84*, 1984.
50. J. Tomida, M. Abe, and T. Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In *ISC 2016*, 2016.
51. B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO 2009*, 2009.

A Additional Definitions

A.1 Function-Hiding Security of DMCFE against Incomplete Queries

Definition 10 (Function-Hiding Security against Incomplete Queries). *For a DMCFE scheme $\mathcal{E}$, a function class $\mathcal{F}$ and a* ppt *adversary $\mathcal{A}$ we define the experiment* $\mathbf{Exp}^{\text{adap-dyn-any-fh-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ *that is the same as the one shown in Figure 1, except that we define a new notion of* admissible adversaries $\mathsf{adm}^{\mathsf{any}}$ *that is given below. The oracles $\mathcal{O}\mathsf{Enc}$, $\mathcal{O}\mathsf{DKeyGen}$ and $\mathcal{O}\mathsf{Corrupt}$ can be called in any order and any number of times. The adversary $\mathcal{A}$ is* NOT admissible *with respect to $\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{DKGen}}$, denoted by $\mathsf{adm}^{\mathsf{any}}(\mathcal{A}) = 0$, if either one of the following holds:*

1. *There exist two distinct tuples of the form $(i, \mathsf{tag}, \cdot, \cdot)$ in $\mathcal{Q}_{\mathsf{Enc}}$ or two distinct tuples of the form $(i, \mathsf{tag\text{-}f}, \cdot, \cdot)$ in $\mathcal{Q}_{\mathsf{DKGen}}$.*
2. *There exists a tuple $(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ such that $i \in \mathcal{C}$ and $x_i^0 \neq x_i^1$, or there exists $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ such that $i \in \mathcal{C}$ and $y_i^0 \neq x_i^1$.*
3. *There exist $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$, two vectors $(x_i^0)_{i\in[n]}, (x_i^1)_{i\in[n]} \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$ and two functions $f^0, f^1 \in \mathcal{F}$ having parameters $(y_i^0, y_i^1)_{i=1}^n$ such that*
   - *$(i, \mathsf{tag}, x_i^0, x_i^1) \in \mathcal{Q}_{\mathsf{Enc}}$ and $(i, \mathsf{tag\text{-}f}, y_i^0, y_i^1) \in \mathcal{Q}_{\mathsf{DKGen}}$ for all $i \in \mathcal{H}$,*
   - *$x_i^0 = x_i^1$ and $y_i^0 = y_i^1$ for all $i \in \mathcal{C}$, and*
   - *$f^0(x_1^0, \ldots, x_n^0) \neq f^1(x_1^1, \ldots, x_n^1)$.*

Otherwise, we say that $\mathcal{A}$ is admissible w.r.t $\mathcal{C}$, $\mathcal{Q}_{\mathsf{Enc}}$ and $\mathcal{Q}_{\mathsf{DKGen}}$ *and write $\mathsf{adm}^{\mathsf{any}}(\mathcal{A}) = 1$. The scheme $\mathcal{E}$ is* function-hiding against incomplete adaptive challenges and dynamic corruption *if for all* ppt *adversaries $\mathcal{A}$,*

$$\mathbf{Adv}^{\text{adap-dyn-any-fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left|\Pr\left[\mathbf{Exp}^{\text{adap-dyn-any-fh-0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\text{adap-dyn-any-fh-1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right]\right|$$

is negligible in λ. Weaker notions can be derived in a similar manner as for Definition 4.

A.2 Pseudorandom Functions

Let families of sets $\mathcal{D} = \{D_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{R} = \{R_\lambda\}_{\lambda\in\mathbb{N}}$ represent domains and ranges. Let $\mathcal{K} = \{K_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of sets where PRF keys are chosen. A *pseudorandom function* is defined by efficient algorithms $\mathsf{PRF} = (\mathsf{KeyGen}, \mathsf{Eval})$, where KeyGen takes as input a security parameter 1^λ and outputs a uniformly random key $K \in \mathcal{K}$; Eval is a deterministic algorithm that inputs a key $K \in \mathcal{K}$ and an element $x \in \mathcal{D}$ to output a range element $z \in \mathcal{R}$.

Security. A pseudorandom function PRF is said to be secure if for sufficiently large $\lambda \in \mathbb{N}$, for all ppt adversary $\mathcal{A}$, the following advantage is negligible in λ:

$$\mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{A}}(1^\lambda) := \left|\Pr\left[\mathsf{Expr}^{\mathsf{PRF}}_{\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2}\right|$$

where $\mathsf{Expr}^{\mathsf{PRF}}_{\mathcal{A}}(1^\lambda)$ is depicted in Fig. 4, $\mathsf{Fun}(\mathcal{D}, \mathcal{R})$ denotes the family of functions going from $\mathcal{D}$ to $\mathcal{R}$, and the probability is taken over the random coins of $\mathcal{A}$.

$\mathsf{Expr}^{\mathsf{PRF}}_{\mathcal{A}}(1^\lambda)$:

$b \xleftarrow{\$} \{0,1\}$
$K \leftarrow \mathsf{KeyGen}(1^\lambda)$
$\mathsf{RF}(\cdot) \xleftarrow{\$} \mathsf{Fun}(\mathcal{D}, \mathcal{R})$
$b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Chal}(\cdot)}(\mathsf{pp})$
Output $b' \stackrel{?}{=} b$

The oracle $\mathcal{O}\mathsf{Chal}$ is defined as follows:

$$\mathcal{O}\mathsf{Chal}(x^*) := \begin{cases} \mathsf{Eval}(K, x^*) \text{ if } b = 1 \\ \mathsf{RF}(x^*) \text{ if } b = 0 \end{cases}$$

Fig. 4: PRF security game.

A.3 IND-CPA Secure Symmetric Encryption

We recall the syntax of symmetric encryption schemes and their *IND-CPA security*. In the transformation, if the key space is larger than the message space we can use one-time pad for simplicity; Otherwise applying a pseudorandom generator to stretch the keys would be suitable as well.
A symmetric key encryption scheme $\mathsf{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ over $\mathcal{K} \times \mathcal{M}$, for a key space $\mathcal{K}$ and a message space $\mathcal{M}$ set up according to a security parameter $\lambda \in \mathbb{N}$, is defined as follows:

- $\mathsf{SEnc}(K, m)$ receives a key $K \in \mathcal{K}$ and a message $m \in \mathcal{M}$, then outputs a ciphertext ct.
- $\mathsf{SDec}(K, \mathsf{ct})$ receives a key $K \in \mathcal{K}$ and a ciphertext ct, then outputs an element in $\mathcal{M}$.

Correctness. For all $K \in \mathcal{K}$ and $m \in \mathcal{M}$, $\Pr[\mathsf{SDec}(K, \mathsf{SEnc}(K, m)) = m] = 1$ where the probability is taken over K.

IND-CPA Security. Let $\mathsf{SE} = (\mathsf{SEnc}, \mathsf{SDec})$ be a symmetric key encryption over $\mathcal{K} \times \mathcal{M}$. Then SE is *IND-secure* if for sufficiently large $\lambda \in \mathbb{N}$ and all ppt adversary $\mathcal{A}$, the following quantity is negligible in λ:

$$\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b : \begin{array}{l} b \xleftarrow{\$} \{0,1\}; K \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow \mathcal{A}^{\mathcal{O}\mathsf{Chal}}(\cdot,\cdot)(1^\lambda) \end{array} \right] - \frac{1}{2} \right| ,$$

where $\mathcal{O}\mathsf{Chal}(m_0, m_1)$ outputs $\mathsf{ct} \leftarrow \mathsf{SEnc}(K, m_b)$, and the probability is taken over the random coins of $\mathcal{A}$.

A.4 Decisional Separation Diffie-Hellman (DSDH) Assumption

Definition 11. *In a cyclic group $\mathbb{G}$ of prime order q, the* **Decisional Separation Diffie-Hellman** (DSDH) *problem is to distinguish the distributions*

$$D_0 = \{(x, y, [\![1]\!], [\![a]\!], [\![b]\!], [\![ab + x]\!])\} \qquad D_1 = \{(x, y, [\![1]\!], [\![a]\!], [\![b]\!], [\![ab + y]\!])\}$$

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.5 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:

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

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

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

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

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

In 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 - Deferred Proofs

B.1 Proof of Lemma 6

Lemma 6. *Let $\mathcal{E} = (\mathsf{Setup},\mathsf{DKeyGen},\mathsf{Enc},\mathsf{Dec})$ be a DMCFE scheme for the function class $\mathcal{F}$. If $\mathcal{E}$ is single-challenge weakly function-hiding (against static corruption), then it is also weakly function-hiding (against static corruption). More specifically, for any* ppt *adversary $\mathcal{A}$, there exists a* ppt *algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \le \max(q_e,q_k)\cdot\mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \ ,$$

where q_e and q_k denote the maximum numbers of different tags tag *and* tag-f *that $\mathcal{A}$ can query to $\mathcal{O}$*Enc *and $\mathcal{O}$*DKeyGen *respectively.*

Proof. For convenience, we introduce slight modifications of the experiments $\mathbf{Exp}^{\mathsf{adap\text{-}stat\text{-}wfh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ and $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, denoted by $\mathbf{Exp}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ and $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$, where the bit b is chosen at random from $\{0,1\}$ during the execution of $\mathsf{Initialize}(1^\lambda)$. Let $\mathcal{A}$ be a ppt adversary in the experiment $\mathbf{Exp}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$. We denote the q_e distinct tags that can occur in a query to $\mathcal{O}\mathsf{Enc}$ by $\mathsf{tag}_1,\ldots,\mathsf{tag}_{q_e}$. Similarly, we denote the q_k tags that can occur in queries to $\mathcal{O}\mathsf{DKeyGen}$ by $\mathsf{tag\text{-}f}_1,\ldots,\mathsf{tag\text{-}f}_{q_k}$. Let $q := \max(q_e,q_k)$. We construct a ppt adversary $\mathcal{B}$ playing against $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ that uses black-box access to $\mathcal{A}$. $\mathcal{B}$ simulates the view of $\mathcal{A}$ as follows:

- *Initialization:* Recall that in the static security model, the set $\mathcal{C} \subset [n]$ of corrupted clients must be declared before the setup.
  Upon $\mathcal{A}$ calling $\mathsf{Initialize}(1^\lambda,\mathcal{C})$, $\mathcal{B}$ samples a random value $j \xleftarrow{\$} [q]$, runs the initialization procedure

$$\mathsf{Initialize}(1^\lambda,\mathcal{C},\mathsf{tag}^* := \mathsf{tag}_j,\mathsf{tag\text{-}f}^* := \mathsf{tag\text{-}f}_j)$$

of its own experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ and forwards the response (consisting of the public parameters pp as well as the encryption keys $\{\mathsf{ek}_i\}_{i\in\mathcal{C}}$ and secret keys $\{\mathsf{sk}_i\}_{i\in\mathcal{C}}$) to $\mathcal{A}$.
Note that if $j > q_e$ (or $j > q_k$), $\mathcal{B}$ submits a challenge tag tag_j (resp. $\mathsf{tag\text{-}f}_j$) that is never part of a query to $\mathcal{O}\mathsf{Enc}$ or $\mathcal{O}\mathsf{DKeyGen}$.

- *Encryption queries:* Recall that tag_ℓ refers to the ℓ-th distinct tag received during an encryption query.
  Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^1_{\ell,i})$, $\mathcal{B}$ behaves as follows:
  1. If $\ell < j$, $\mathcal{B}$ queries the oracle $\mathcal{O}\mathsf{Enc}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ on input $(i, \mathsf{tag}_\ell, x^1_i, x^1_i)$ and forwards the response to $\mathcal{A}$.
  2. If $\ell = j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^1_{\ell,i})$ and forwards the response to $\mathcal{A}$.
  3. If $\ell > j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^0_{\ell,i})$ and forwards the response to $\mathcal{A}$.
- *Key-generation queries:* Recall also that $\mathsf{tag\text{-}f}_k$ refers to the k-th distinct tag received during a key-generation query.
  Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}$ on input $(i, \mathsf{tag}_k, y^0_{k,i}, y^1_{k,i})$, $\mathcal{B}$ does the following:
  1. If $k < j$, $\mathcal{B}$ queries the oracle $\mathcal{O}\mathsf{DKeyGen}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ on input $(i, \mathsf{tag\text{-}f}_k, y^1_{k,i}, y^1_{k,i})$ and forwards the response to $\mathcal{A}$.
  2. If $k = j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^1_{k,i})$ and forwards the response to $\mathcal{A}$.
  3. If $k > j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^0_{k,i})$ and forwards the response to $\mathcal{A}$.
- *Finalize:* Upon a calling $\mathsf{Finalize}(b')$, $\mathcal{B}$ passes the same bit b' to its own $\mathsf{Finalize}$ procedure.

We note that thanks to the weakly function-hiding setting, $\mathcal{A}$ is an admissible adversary against $\mathbf{Exp}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ if and only if $\mathcal{B}$ is an admissible adversary against $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$.
We define a sequence of hybrid games $\mathcal{H}_0, \mathcal{H}_1, \ldots, \mathcal{H}_q$ where we modify the definition of the oracles $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{DKeyGen}$ in $\mathbf{Exp}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$. Specifically, in $\mathcal{H}_j$ a query $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^1_{\ell,i})$ is answered by an encryption of $x^1_{\ell,i}$ if $\ell \le j$ and by an encryption of $x^0_{\ell,i}$ if $\ell > j$. Similarly, a query $(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^1_{k,i})$ to $\mathcal{O}\mathsf{DKeyGen}$ is answered by a partial decryption key for $y^1_{k,i}$ if $k \le j$ and by a partial decryption key for $y^0_{k,i}$ if $k > j$. We denote by $\mathcal{H}_j = 1$ the event where the hybrid game outputs 1. Observe that $\mathbf{Adv}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = |\Pr[\mathcal{H}_q = 1] - \Pr[\mathcal{H}_0 = 1]|$.
The advantage of $\mathcal{B}$ against the single-challenge weakly function-hiding experiment of $\mathcal{E}$ is

$$
\begin{aligned}
&\mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \\
&= \Big| \Pr\Big[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) = 1 \mid b = 0\Big] \\
&\quad - \Pr\Big[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) = 1 \mid b = 1\Big] \Big| \\
&= \frac{1}{q} \cdot \Bigg| \sum_{j=1}^{q} \Big(\Pr\Big[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) = 1 \mid b = 0, \mathcal{B} \text{ picks } j\Big] \\
&\quad - \Pr\Big[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) = 1 \mid b = 1, \mathcal{B} \text{ picks } j\Big] \Big) \Bigg| \\
&\overset{(*)}{\ge} \frac{1}{q} \cdot \left| \sum_{j=1}^{q} \left(\Pr\left[\mathcal{H}_j = 1\right] - \Pr\left[\mathcal{H}_{j-1} = 1\right] \right) \right| \\
&= \frac{1}{q} \cdot \left| \Pr\left[\mathcal{H}_q = 1\right] - \Pr\left[\mathcal{H}_0 = 1\right] \right| \\
&= \frac{1}{q} \cdot \mathbf{Adv}^{\mathsf{adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)
\end{aligned}
$$

where $(*)$ comes from the observation that conditioned on $[b=0]$ (resp. $[b=1]$) and $[\mathcal{B}$ picks $j]$, $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)=1$ is identical to $\mathcal{H}_j=1$ (resp. $\mathcal{H}_{j-1}=1$), where $\mathcal{B}$ is simulating the challenger for $\mathcal{A}$. □

B.2 From Weak to Full Function-Hiding

Lemma 7. *Let $\mathcal{F}^{\mathsf{IP}}_n$ denote the function class containing inner products of length n and let $\mathcal{E}$ be a weakly function-hiding DMCFE scheme for $\mathcal{F}^{\mathsf{IP}}_{2n}$. Then there exists a (fully) function-hiding DMCFE scheme $\mathcal{E}'$ for $\mathcal{F}^{\mathsf{IP}}_n$. More precisely, for any* ppt *adversary $\mathcal{A}$, there exists a* ppt *algorithm $\mathcal{B}$ such that*

$$\mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}fh}}_{\mathcal{E}',\mathcal{F}^{\mathsf{IP}}_n,\mathcal{A}}(1^\lambda)\le 3\cdot\mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{IP}}_{2n},\mathcal{B}}(1^\lambda)\ ,$$

where $\mathsf{xxx}\in\{\mathsf{adap},\mathsf{sel}\}$ *and* $\mathsf{yyy}\in\{\mathsf{dyn},\mathsf{stat}\}$.

Proof. Let $\mathcal{E}=(\mathsf{Setup},\mathsf{DKeyGen},\mathsf{Enc},\mathsf{Dec})$. Then we define the DMCFE scheme $\mathcal{E}'=(\mathsf{Setup}',\mathsf{Enc}',\mathsf{DKeyGen}',\mathsf{Dec}')$ for $\mathcal{F}^{\mathsf{IP}}_n$ as follows:

- *Setup:* $\mathsf{Setup}'(1^\lambda)$ runs

$$(\mathsf{pp},\{\mathsf{sk}_i,\mathsf{ek}_i\}_{i\in[2n]})\leftarrow\mathsf{Setup}(1^\lambda)$$

  and outputs $(\mathsf{pp}'=\mathsf{pp},\{\mathsf{sk}'_i=(\mathsf{sk}_i,\mathsf{sk}_{n+i}),\mathsf{ek}'_i=(\mathsf{ek}_i,\mathsf{ek}_{n+i})\})$.
- *Key Generation:* $\mathsf{DKeyGen}'(\mathsf{sk}'_i,\mathsf{tag\text{-}f},y_i)$ parses $\mathsf{sk}'_i=(\mathsf{sk}_i,\mathsf{sk}_{n+i})$ and runs

$$\begin{aligned}\mathsf{dk}_{\mathsf{tag\text{-}f},i}&\leftarrow\mathsf{DKeyGen}(\mathsf{sk}_i,\mathsf{tag\text{-}f},y_i)\\ \mathsf{dk}_{\mathsf{tag\text{-}f},n+i}&\leftarrow\mathsf{DKeyGen}(\mathsf{sk}_{n+i},\mathsf{tag\text{-}f},0)\ .\end{aligned}$$

  Finally, it outputs $\mathsf{dk}'_{\mathsf{tag\text{-}f},i}:=(\mathsf{dk}_{\mathsf{tag\text{-}f},i},\mathsf{dk}_{\mathsf{tag\text{-}f},n+i})$.
- *Encryption:* $\mathsf{Enc}'(\mathsf{ek}'_i,\mathsf{tag},x_i)$ parses $\mathsf{ek}'_i=(\mathsf{ek}_i,\mathsf{ek}_{n+i})$ and runs

$$\begin{aligned}\mathsf{ct}_{\mathsf{tag},i}&\leftarrow\mathsf{Enc}(\mathsf{ek}_i,\mathsf{tag},x_i)\\ \mathsf{ct}_{\mathsf{tag},n+i}&\leftarrow\mathsf{Enc}(\mathsf{ek}_{n+i},\mathsf{tag},0)\ .\end{aligned}$$

  Finally, it outputs $\mathsf{ct}'_{\mathsf{tag},i}:=(\mathsf{ct}_{\mathsf{tag},i},\mathsf{ct}_{\mathsf{tag},n+i})$.
- *Decryption:* $\mathsf{Dec}((\mathsf{dk}'_{\mathsf{tag\text{-}f},i})_{i\in[n]},(\mathsf{ct}'_{\mathsf{tag},i})_{i\in[n]})$ parses the n decryption keys and ciphertexts

$$\begin{aligned}\mathsf{dk}'_{\mathsf{tag\text{-}f},i}&:=(\mathsf{dk}_{\mathsf{tag\text{-}f},i},\mathsf{dk}_{\mathsf{tag\text{-}f},n+i})\\ \mathsf{ct}'_{\mathsf{tag},i}&:=(\mathsf{ct}_{\mathsf{tag},i},\mathsf{ct}_{\mathsf{tag},n+i})\ .\end{aligned}$$

  Finally, it outputs $d\leftarrow\mathsf{Dec}((\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i\in[2n]},(\mathsf{ct}_{\mathsf{tag},i})_{i\in[2n]})$.

The correctness of $\mathcal{E}'$ follows immediately from that of $\mathcal{E}$ and the fact that $\langle(\mathbf{x}\parallel\mathbf{0}),(\mathbf{y}\parallel\mathbf{0})\rangle=\langle\mathbf{x},\mathbf{y}\rangle$ for $\mathbf{x}=(x_i)_{i\in[n]}$, $\mathbf{y}=(y_i)_{i\in[n]}$ and $\mathbf{0}=(0,\dots,0)\in\mathbb{Z}_q^n$.
Furthermore, we show that $\mathcal{E}'$ enjoys the (full-fledged) function-hiding property. Towards this, we consider a sequence of hybrid games $\mathsf{G}_0,\dots,\mathsf{G}_3$ where G_0 equals $\mathbf{Exp}^{\mathsf{xxx\text{-}yyy\text{-}fh\text{-}0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ and G_3 equals $\mathbf{Exp}^{\mathsf{xxx\text{-}yyy\text{-}fh\text{-}1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

Game G_0: This is $\mathbf{Exp}^{\mathsf{xxx\text{-}yyy\text{-}fh\text{-}0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$. We denote the ℓ-th distinct tag that occurs in a query to $\mathcal{O}\mathsf{Enc}$ by tag_ℓ. Similarly, $\mathsf{tag\text{-}f}_k$ refers to the k-th distinct tag in a query to $\mathcal{O}\mathsf{DKeyGen}$. Queries to $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{DKeyGen}$ are answered as follows:

- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^1_{\ell,i})$, the challenger computes

$$\begin{aligned}\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}_\ell, x^0_{\ell,i})\\ \mathsf{ct}_{\ell,n+i} &\leftarrow \mathsf{Enc}(\mathsf{ek}_{n+i}, \mathsf{tag}_\ell, 0)\ .\end{aligned}$$

  Then it returns $\mathsf{ct}'_{\ell,i} \coloneqq (\mathsf{ct}_{\ell,i}, \mathsf{ct}_{\ell,n+i})$.
- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^1_{k,i})$, the challenger computes

$$\begin{aligned}\mathsf{dk}_{k,i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}_k, y^0_{k,i})\\ \mathsf{dk}_{k,n+i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_{n+i}, \mathsf{tag\text{-}f}_k, 0)\ .\end{aligned}$$

  Then it returns $\mathsf{dk}'_{k,i} \coloneqq (\mathsf{dk}_{k,i}, \mathsf{dk}_{k,n+i})$.

In other words, the ciphertexts $(\mathsf{ct}'_{\ell,i})_{i\in[n]}$ encrypt the vector $(\mathbf{x}^0_\ell \parallel \mathbf{0})$, and the partial decryption keys $(\mathsf{dk}'_{k,i})_{i\in[n]}$ allow for the computation of the inner product with the vector $(\mathbf{y}^0_k \parallel \mathbf{0})$.

Game G_1: We modify the definition of $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{DKeyGen}$ as follows:

- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^1_{\ell,i})$, the challenger computes

$$\begin{aligned}\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}_\ell, 0)\\ \mathsf{ct}_{\ell,n+i} &\leftarrow \mathsf{Enc}(\mathsf{ek}_{n+i}, \mathsf{tag}_\ell, x^1_{\ell,i})\ .\end{aligned}$$

  Then it returns $\mathsf{ct}'_{\ell,i} \coloneqq (\mathsf{ct}_{\ell,i}, \mathsf{ct}_{\ell,n+i})$.
- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^1_{k,i})$, the challenger computes

$$\begin{aligned}\mathsf{dk}_{k,i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}_k, y^0_{k,i})\\ \mathsf{dk}_{k,n+i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_{n+i}, \mathsf{tag\text{-}f}_k, y^1_{k,i})\ .\end{aligned}$$

  Then it returns $\mathsf{dk}'_{k,i} \coloneqq (\mathsf{dk}_{k,i}, \mathsf{dk}_{k,n+i})$.

Thus, the ciphertexts $(\mathsf{ct}'_{\ell,i})_{i\in[n]}$ encrypt the vector $(\mathbf{0} \parallel \mathbf{x}^1_\ell)$ (as opposed to $(\mathbf{x}^0_\ell \parallel \mathbf{0})$ in G_0), and the partial decryption keys $(\mathsf{dk}'_{k,i})_{i\in[n]}$ allow for the computation of the inner product with the vector $(\mathbf{y}^0_k \parallel \mathbf{y}^1_k)$ (as opposed to $(\mathbf{y}^0_k, \mathbf{0})$ in G_0). The admissibility of $\mathcal{A}$ states that $\langle \mathbf{x}^0_\ell, \mathbf{y}^0_k\rangle = \langle \mathbf{x}^1_\ell, \mathbf{y}^1_k\rangle$ which implies that

$$\langle \mathbf{x}^0_\ell \parallel \mathbf{0}, \mathbf{y}^0_k \parallel \mathbf{0}\rangle = \langle \mathbf{x}^0_\ell \parallel \mathbf{0}, \mathbf{y}^0_k \parallel \mathbf{y}^1_k\rangle = \langle \mathbf{0} \parallel \mathbf{x}^1_\ell, \mathbf{y}^0_k \parallel \mathbf{y}^1_k\rangle\ .$$

Then it follows by the weak function-hiding property of $\mathcal{E}'$ that there exists a ppt adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1]| \le \mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{IP}}_{2n},\mathcal{B}}(1^\lambda)$.

Game G_2: We modify the definition of $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{DKeyGen}$ again.

- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}_\ell, x^0_{\ell,i}, x^1_{\ell,i})$, the challenger computes

$$\begin{aligned}\mathsf{ct}_{\ell,i} &\leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}_\ell, x^1_{\ell,i})\\ \mathsf{ct}_{\ell,n+i} &\leftarrow \mathsf{Enc}(\mathsf{ek}_{n+i}, \mathsf{tag}_\ell, 0)\ .\end{aligned}$$

  Then it returns $\mathsf{ct}'_{\ell,i} \coloneqq (\mathsf{ct}_{\ell,i}, \mathsf{ct}_{\ell,n+i})$.
- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^1_{k,i})$, the challenger computes

$$\begin{aligned}\mathsf{dk}_{k,i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}_k, y^1_{k,i})\\ \mathsf{dk}_{k,n+i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_{n+i}, \mathsf{tag\text{-}f}_k, y^1_{k,i})\ .\end{aligned}$$

  Then it returns $\mathsf{dk}'_{k,i} \coloneqq (\mathsf{dk}_{k,i}, \mathsf{dk}_{k,n+i})$.

That is, the challenger provides a ciphertext of $(\mathbf{x}^1_\ell \parallel \mathbf{0})$ and a decryption key for $(\mathbf{y}^1_k \parallel \mathbf{y}^1_k)$, as opposed to $(\mathbf{0} \parallel \mathbf{x}^1_\ell)$ and $(\mathbf{y}^0_k \parallel \mathbf{y}^1_k)$ in G_1. Notice that

$$\langle \mathbf{0} \parallel \mathbf{x}^1_\ell, \mathbf{y}^0_k \parallel \mathbf{y}^1_k \rangle = \langle \mathbf{0} \parallel \mathbf{x}^1_\ell, \mathbf{y}^1_k \parallel \mathbf{y}^1_k \rangle = \langle \mathbf{x}^1_\ell \parallel \mathbf{0}, \mathbf{y}^1_k \parallel \mathbf{y}^1_k \rangle \ .$$

Then it follows by the weak function-hiding property of $\mathcal{E}'$ that there exists a ppt adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{IP}}_{2n},\mathcal{B}}(1^\lambda)$.

Game G_3: We modify the definition of $\mathcal{O}\mathsf{DKeyGen}$ as follows. (The definition of $\mathcal{O}\mathsf{Enc}$ is as in G_2.)

- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}_k, y^0_{k,i}, y^1_{k,i})$, the challenger computes

$$\begin{aligned} \mathsf{dk}_{k,i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}_k, y^1_{k,i}) \\ \mathsf{dk}_{k,n+i} &\leftarrow \mathsf{DKeyGen}(\mathsf{sk}_{n+i}, \mathsf{tag\text{-}f}_k, 0) \ . \end{aligned}$$

  Then it returns $\mathsf{dk}'_{k,i} := (\mathsf{dk}_{k,i}, \mathsf{dk}_{k,n+i})$.

Thus, the challenger provides a decryption key for $(\mathbf{y}^1_k \parallel \mathbf{0})$, as opposed to $(\mathbf{y}^1_k \parallel \mathbf{y}^1_k)$ in G_2. We have

$$\langle \mathbf{x}^1_\ell \parallel \mathbf{0}, \mathbf{y}^1_k \parallel \mathbf{y}^1_k \rangle = \langle \mathbf{x}^1_\ell \parallel \mathbf{0}, \mathbf{y}^1_k \parallel \mathbf{0} \rangle \ .$$

As above, it follows by the weak function-hiding property of $\mathcal{E}'$ that there exists a ppt adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq \mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{IP}}_{2n},\mathcal{B}}(1^\lambda)$. Note that G_3 equals the experiment $\mathbf{Exp}^{\mathsf{xxx\text{-}yyy\text{-}fh\text{-}1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

Therefore, by a hybrid argument, we conclude that

$$\begin{aligned} &\left| \Pr\left[\mathbf{Exp}^{\mathsf{xxx\text{-}yyy\text{-}fh\text{-}0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right] - \Pr\left[\mathbf{Exp}^{\mathsf{xxx\text{-}yyy\text{-}fh\text{-}1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right] \right| \\ &\quad \leq 3 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}yyy\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{IP}}_{2n},\mathcal{B}}(1^\lambda) \ . \end{aligned}$$

B.3 Proof of Lemma 8

Lemma 8 (Secret-Sharing Swapping). *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda) \in \mathbb{N}$ where $H, K, L : \mathbb{N} \to \mathbb{N}$ are functions. Let $(\mathbf{B}_i, \mathbf{B}^*_i)_{i\in[H]}$ be two random dual bases of dimension 9 in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$. All basis vectors are kept secret.*
We consider any public values $(x_i, x_{\ell,i}, x'_{\ell,i}, y^0_i, y^1_i, y_{k,i}, R, R_k)_{i\in[H],k\in[K],\ell\in[L]}$ such that $\sum_{i=1}^H x_i y^0_i = \sum_{i=1}^H x_i y^1_i$. With random $r, r_\ell, \rho_i, \rho_{\ell,i}, \pi_i, \pi_{k,i}, \sigma_i, \sigma_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ such that $\sum_{i=1}^H \sigma_i = R$ and $\sum_{i=1}^H \sigma_{k,i} = R_k$, for all $k \in [K]$, the following distributions are computationally indistinguishable under the SXDH *assumption:*

$$D_0 := \left\{ \begin{array}{l} (\mathbf{c}_{\ell,i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i})_{\ell\in[L]} \\ \left(\mathbf{c}_i = (\boxed{x_i},\ \boxed{0},\ r,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}\right)_{i\in[H]} \\ \left(\mathbf{d}_i = (y^1_i,\ y^0_i,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i}\right)_{i\in[H]} \\ \left(\mathbf{d}_{k,i} = (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i}\right)_{i\in[H],k\in[K]} \end{array} \right\};$$

$$D_1 := \left\{ \begin{array}{l} (\mathbf{c}_{\ell,i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i})_{\ell\in[L]} \\ \left(\mathbf{c}_i = (\boxed{0},\ \boxed{x_i},\ r,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}\right)_{i\in[H]} \\ \left(\mathbf{d}_i = (y^1_i,\ y^0_i,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i}\right)_{i\in[H]} \\ \left(\mathbf{d}_{k,i} = (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i}\right)_{i\in[H],k\in[K]} \end{array} \right\} \ .$$

More specifically, we have:

$$\left| \Pr_{\mathsf{samp}\sim D_0}[\mathcal{A}(\mathsf{samp})\to 1] - \Pr_{\mathsf{samp}\sim D_1}[\mathcal{A}(\mathsf{samp})\to 1] \right| \leq 12\cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

for any specific ppt $\mathcal{A}$ *with fixed* $(R,(x_i,(x_{\ell,i},x'_{\ell,i})_{\ell=1}^L,y_i^0,y_i^1,(y_{k,i},R_k)_{k=1}^K)_{i=1}^H)$.

Proof (of Lemma 8). The sequence of games is given in Figure 2. The changes that make the transition between games are highlighted by a frame. In the vectors, we write 0^t to denote t consecutive coordinates containing 0. The details of the transition are given as follows:

Game G_0: The vectors are sampled according to D_0.

Game G_1: We perform a computational basis change, making use of the randomness $r \xleftarrow{\$} \mathbb{Z}_q$ at coordinate 3 of $(\mathbf{c}_i)_{i=1}^H$ and of $\sigma_i \xleftarrow{\$} \mathbb{Z}_q$ at coordinate 3 of $(\mathbf{d}_i)_{i=1}^H$ so as to introduce a new non-zero $r' \xleftarrow{\$} \mathbb{Z}_q^*$ and secret sharing $(\tau_i)_{i=1}^H$ of 0 with only non-zero τ_i, at coordinate 9 in $(\mathbf{c}_i)_{i=1}^H$ and $(\mathbf{d}_i)_{i=1}^H$. We recall that it holds $\sum_{i=1}^H \sigma_i = R$ for some fixed public value R. We proceed in two steps:

Game $\mathsf{G}_{0.1}$: We first use the subspace-indistinguishability to introduce $r' \xleftarrow{\$} \mathbb{Z}_q^*$ at coordinate 9 of $\mathbf{c}_i$, while keeping $\mathbf{d}_i[9] = \mathbf{c}_{\ell,i}[9] = \mathbf{d}_{k,i}[9] = 0$. Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta \coloneqq c - ab$ is either 0 or 1, the basis changing matrices are:

$$\mathbf{B}_i = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{3,9} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{3,9} \cdot \mathbf{H}_i^* \ .$$

All vectors changed under these bases are secret. We compute $\mathbf{B}_i$ using $[\![a]\!]_1$ and write the $\mathbf{c}$-vectors as follows:

$$\begin{aligned} \mathbf{c}_i &= (x_i,\ 0,\ r,\ 0,\ \rho_i,\ 0^3,\ 0)_{\mathbf{B}_i} + (0,\ 0,\ br',\ 0,\ 0,\ 0^3,\ cr')_{\mathbf{H}_i} \\ &= (x_i,\ 0, \boxed{r + br'},\ 0,\ \rho_i,\ 0^3, \boxed{\delta r'})_{\mathbf{B}_i} \\ \mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^3,\ 0)_{\mathbf{B}_i} \ . \end{aligned}$$

We cannot compute $\mathbf{b}_{i,3}^*$ but can write the $\mathbf{d}$-vectors in $\mathbf{H}^*$ and observe that they stay invariant in $\mathbf{B}_i^*$ as the 9-th coordinate is 0:

$$\begin{aligned} \mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0^5)_{\mathbf{H}_i^*} = (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0^5)_{\mathbf{B}_i^*} \\ \mathbf{d}_{k,i} &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0^5)_{\mathbf{H}_i^*} = (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0^5)_{\mathbf{B}_i^*} \ . \end{aligned}$$

If $\delta = 0$ we are in G_0 else we are in $\mathsf{G}_{0.1}$, while updating r to $r + br'$. The difference in advantages is $|\Pr[\mathsf{G}_{0.1} = 1] - \Pr[\mathsf{G}_0 = 1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

Game $\mathsf{G}_{0.2}$: We use DSDH in $\mathbb{G}_2$ to introduce any chosen secret sharing $(\tau_i)_{i\in[H]}$ of 0, *i.e.* $\sum_{i=1}^H \tau_i = 0$, such that $\tau_i \neq 0$ for all i. Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta \coloneqq c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{3,9} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{3,9} \cdot \mathbf{H}_i^* \ .$$

All vectors changed under these bases are secret. We compute $\mathbf{B}_i^*$ using $[\![a]\!]_2$ and write the $\mathbf{d}$-vectors as follows:

$$\begin{aligned}\mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0^4,\ 0)_{\mathbf{B}_i^*} + (0,\ 0,\ b\tau_i,\ 0,\ 0^4,\ c\tau_i)_{\mathbf{H}_i^*}\\ &= (y_i^1,\ y_i^0, \boxed{\sigma_i + b\tau_i},\ \pi_i,\ 0^4, \boxed{\delta\tau_i})_{\mathbf{B}_i^*}\\ \mathbf{d}_{k,i} &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0^4,\ 0)_{\mathbf{B}_i}\ .\end{aligned}$$

The secret shares $(\sigma_i)_{i=1}^H$ are updated to $(\sigma_i + b\tau_i)_{i=1}^H$ and still satisfy:

$$\sum_{i=1}^{H}(\sigma_i + b\tau_i) = \left(\sum_{i=1}^{H}\sigma_i\right) + b\left(\sum_{i=1}^{H}\tau_i\right) = R\ .$$

We cannot compute $\mathbf{b}_{i,9}$ but can write the $\mathbf{c}$-vectors in $\mathbf{H}$, for $r'', r_\ell \xleftarrow{\$} \mathbb{Z}_q$, $r' \xleftarrow{\$} \mathbb{Z}_q^*$:

$$\begin{aligned}\mathbf{c}_i &= (x_i,\ 0,\ r'',\ 0,\ \rho_i,\ 0^3,\ r')_{\mathbf{H}_i} = (x_i,\ 0,\ r'' + ar',\ 0,\ \rho_i,\ 0^3,\ r')_{\mathbf{B}_i}\\ \mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^3,\ 0)_{\mathbf{H}_i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^3,\ 0)_{\mathbf{B}_i}\ ,\end{aligned}$$

while simulating $r \coloneqq r'' + ar'$ perfectly uniformly at random in $\mathbb{Z}_q$. If $\delta = 0$ we are in $\mathsf{G}_{0.1}$, else we are in $\mathsf{G}_{0.2} = \mathsf{G}_1$. The difference in advantages is $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_{0.1} = 1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

After $\mathsf{G}_{0.2} = \mathsf{G}_1$, the vectors are now:

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^3,\ 0)_{\mathbf{B}_i};\quad \mathbf{c}_i = (x_i,\ 0,\ \boxed{r},\ 0,\ \rho_i,\ 0^3,\ \boxed{r'})_{\mathbf{B}_i}\\ \mathbf{d}_i &= (y_i^1,\ y_i^0,\ \boxed{\sigma_i},\ \pi_i,\ 0,\ 0^3,\ \boxed{\tau_i})_{\mathbf{B}_i^*};\quad \mathbf{d}_{k,i} = (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0^3, 0)_{\mathbf{B}_i^*}\end{aligned}$$

and in total $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

Game G_2: We perform a formal basis change to duplicate (y_i^1, y_i^0) (respectively $(y_{k,i}, y_{k,i})$) from coordinates $(1,2)$ to coordinates $(7,8)$ of $\mathbf{d}_i$ (respectively of $\mathbf{d}_{k,i}$). The bases are changed following:

$$\mathbf{B}_i = \begin{bmatrix}1&0&0&0\\0&1&0&0\\1&0&1&0\\0&1&0&1\end{bmatrix}_{1,2,7,8} \cdot \mathbf{H}_i;\quad \mathbf{B}_i^* = \begin{bmatrix}1&0&-1&0\\0&1&0&-1\\0&0&1&0\\0&0&0&1\end{bmatrix}_{1,2,7,8} \cdot \mathbf{H}_i^*\ .$$

We write the vectors as follows, observing that the $\mathbf{c}$-vectors stay invariant because their coordinates $(7,8)$ are 0 and the duplication is done correctly for the $\mathbf{d}$-vectors:

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0^2,\ 0)_{\mathbf{H}_i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0^2,\ 0)_{\mathbf{B}_i}\\ \mathbf{c}_i &= (x_i,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0^2,\ r')_{\mathbf{H}_i} = (x_i,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0^2,\ r')_{\mathbf{B}_i}\\ \mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0^4,\ \tau_i)_{\mathbf{H}_i^*} = (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0^2,\ \boxed{y_i^1,\ y_i^0},\ \tau_i)_{\mathbf{B}_i^*}\\ \mathbf{d}_{k,i} &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0^4,\ 0)_{\mathbf{H}_i^*} = (y_{k,i}, y_{k,i}, \sigma_{k,i}, \pi_{k,i}, 0^2, \boxed{y_{k,i}, y_{k,i}}, 0)_{\mathbf{B}_i^*}\ .\end{aligned}$$

We are in G_1 in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ and in G_2 in bases $(\mathbf{B}_i, \mathbf{B}_i^*)$. The change is formal and we have $\Pr[\mathsf{G}_2 = 1] = \Pr[\mathsf{G}_1 = 1]$.

Game G_3: We perform a computational change to swap x_i from coordinate 1 to coordinate 7 of $\mathbf{c}_i$. Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta := c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 & 0 \\ -a & 1 & a \\ 0 & 0 & 1 \end{bmatrix}_{1,5,7} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & -a & 1 \end{bmatrix}_{1,5,7} \cdot \mathbf{H}_i^* \ .$$

All vectors changed under these bases are secret. We compute $\mathbf{B}_i$ using $[\![a]\!]_1$ and write the $\mathbf{c}$-vectors as follows:

$$\begin{aligned} \mathbf{c}_i &= (x_i,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ r')_{\mathbf{B}_i} + (-cx_i,\ 0,\ 0,\ 0,\ bx_i,\ 0,\ cx_i,\ 0,\ 0)_{\mathbf{H}_i} \\ &= (\boxed{x_i - \delta x_i},\ 0,\ r,\ 0, \boxed{\rho_i + bx_i},\ 0, \boxed{\delta x_i},\ 0,\ r')_{\mathbf{B}_i} \\ \mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i} \ . \end{aligned}$$

We cannot compute $\mathbf{b}_{i,1}^*$ and $\mathbf{b}_{1,7}^*$ due to the lack of $[\![a]\!]_2$, but the $\mathbf{d}$-vectors can be written in $\mathbf{H}_i^*$ indeed they stay invariant: for instance we consider $\mathbf{d}_i$, the same holds for $\mathbf{d}_{k,i}$

$$\begin{aligned} \mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ y_i^1,\ y_i^0,\ \tau_i)_{\mathbf{H}_i^*} \\ &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i, -ay_i^1 + ay_i^1,\ 0,\ y_i^1,\ y_i^0,\ \tau_i)_{\mathbf{B}_i^*} \\ &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ y_i^1,\ y_i^0,\ \tau_i)_{\mathbf{B}_i^*} \ . \end{aligned}$$

If $\delta = 0$ we are in G_2, else we are in G_3, while updating ρ_i to $\rho_i + bx_i$. We have $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

Game G_4: We perform a formal duplication to have a copy of x_i in coordinate 6 from coordinate 7 of $\mathbf{d}_i$. The bases are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}_{6,7} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}_{6,7} \cdot \mathbf{H}_i^* \ .$$

All vectors are kept secret. We write the $\mathbf{c}$-vectors to observe the duplication:

$$\begin{aligned} \mathbf{c}_i &= (0,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ x_i,\ 0,\ r')_{\mathbf{H}_i} = (0,\ 0,\ r,\ 0,\ \rho_i,\ \boxed{x_i},\ x_i,\ 0,\ r')_{\mathbf{B}_i} \\ \mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{H}_i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \ , \end{aligned}$$

where $\mathbf{c}_{\ell,i}$ stays invariant because $\mathbf{c}_{\ell,i}[7] = 0$. In the dual bases, $\mathbf{d}_i[6] = \mathbf{d}_{k,i}[6] = 0$ means the writings of $\mathbf{d}_i, \mathbf{d}_{k,i}$ are the same in $\mathbf{H}^*$ as in $\mathbf{B}_i^*$. We are in G_3 in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ and in G_4 in bases $(\mathbf{B}_i, \mathbf{B}_i^*)$. The change is formal and we have $\Pr[\mathsf{G}_4 = 1] = \Pr[\mathsf{G}_3 = 1]$.

Game G_5 : We perform a formal change, using formal basis changes to swap x_i from coordinate 7 to coordinate 8 in $\mathbf{c}_i$. The bases are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 1 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{6,7,8} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}_{6,7,8} \cdot \mathbf{H}_i^* \ .$$

We write the $\mathbf{c}$-vectors to observe the duplication:

$$\begin{aligned}\mathbf{c}_i &= (0^2, r, 0, \rho_i, x_i, x_i, 0, r')_{\mathbf{H}_i} = (0^2, r, 0, \rho_i, x_i, \boxed{x_i - x_i, x_i}, r')_{\mathbf{B}_i}\\ &= (0^2, r, 0, \rho_i, x_i, \boxed{0, x_i}, r')_{\mathbf{B}_i}\\ \mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{H}_i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\ ,\end{aligned}$$

where $\mathbf{c}_{\ell,i}$ stays invariant because its 6-th coordinate is 0. Moreover, in the dual basis, the $\mathbf{d}$-vectors change as follows:

$$\begin{aligned}\mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ y_i^1,\ y_i^0,\ \tau_i)_{\mathbf{H}_i^*}\\ &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ \boxed{y_i^1 - y_i^0},\ y_i^1,\ y_i^0,\ \tau_i)_{\mathbf{H}_i^*}\\ \mathbf{d}_{k,i} &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0,\ y_{k,i},\ y_{k,i},\ 0)_{\mathbf{H}_i^*}\\ &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ y_{k,i} - y_{k,i},\ y_{k,i},\ y_{k,i},\ 0)_{\mathbf{B}_i^*}\\ &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ \boxed{0},\ y_{k,i},\ y_{k,i},\ 0)_{\mathbf{B}_i^*}\ .\end{aligned}$$

We are in G_4 in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ and in G_5 in bases $(\mathbf{B}_i, \mathbf{B}_i^*)$. The change is formal and we have $\Pr[\mathsf{G}_5 = 1] = \Pr[\mathsf{G}_4 = 1]$.

The vectors, when we arrive at G_5, are of the form:

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}; \mathbf{c}_i = (0,\ 0,\ r,\ 0,\ \rho_i,\ x_i,\ 0,\ x_i,\ r')_{\mathbf{B}_i}\\ \mathbf{d}_i &= (y_i^1, y_i^0, \sigma_i, \pi_i, 0, \Delta y_i, y_i^1, y_i^0, \tau_i)_{\mathbf{B}_i^*}; \mathbf{d}_{k,i} = (y_{k,i}, y_{k,i}, \sigma_{k,i}, \pi_{k,i}, 0, 0, y_{k,i}, y_{k,i}, 0)_{\mathbf{B}_i^*}\end{aligned}$$

where $\Delta y_i \coloneqq y_i^1 - y_i^0$, $(\tau_i)_{i=1}^H$ is a random secret sharing of 0, with $\tau_i \neq 0$ for all i, and $r' \xleftarrow{\$} \mathbb{Z}_q^*$. Our goal in the next three games $\mathsf{G}_6, \mathsf{G}_7, \mathsf{G}_8$ is to clean coordinate 6 of $\mathbf{c}_i, \mathbf{d}_i$. The main idea is to consider the *selective* version G_j^* for $j \in \{5, 6, 7, 8\}$, where the values $(x_i, y_i^0, y_i^1)_{i\in[H]}$ are guessed in advance. We then use formal argument for the transitions $\mathsf{G}_j^* \to \mathsf{G}_{j+1}^*$ for $j \in \{5, 6, 7\}$ to obtain

$$\Pr[\mathsf{G}_5^* = 1] = \Pr[\mathsf{G}_6^* = 1] = \Pr[\mathsf{G}_7^* = 1] = \Pr[\mathsf{G}_8^* = 1]\ . \tag{5}$$

In the end, we use a *complexity leveraging* argument to conclude that thanks to (5), we have $\Pr[\mathsf{G}_5 = 1] = \Pr[\mathsf{G}_6 = 1] = \Pr[\mathsf{G}_7 = 1] = \Pr[\mathsf{G}_8 = 1]$. For the sequence $\mathsf{G}_5 \to \mathsf{G}_8$, we make a guess for the values $(x_i, y_i^0, y_i^1)_{i\in[H]}$, choose $r' \xleftarrow{\$} \mathbb{Z}_q^*$, two random secret sharings $(\tau_i)_{i=1}^H$, $(\tau'_i)_{i=1}^H$ of 0, with $\tau_i, \tau'_i \neq 0$ for all i, and define the event E that the guess is correct on the values $(x_i, y_i^0, y_i^1)_{i\in[H]}$ and $\tau'_i - \tau_i = x_i \Delta y_i / r'$, where $\Delta y_i \coloneqq y_i^1 - y_i^0$. We describe the selective games below, starting from G_5^*, where event E is assumed true:

Game G_6^* : Knowing $(x_i, y_i^1, y_i^0)_{i=1}^H$ in advance, we perform a formal quotient on coordinate 6 of $\mathbf{c}_i$ and of $\mathbf{d}_i$. Without loss of generality we can assume $x_i \neq 0$, otherwise the vector $\mathbf{c}_i$ is trivially belonging to the distribution D_1 since G_0. The bases are changed following:

$$\mathbf{B}_i = \left[\tfrac{r'}{x_i}\right]_6 \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \left[\tfrac{x_i}{r'}\right]_6 \cdot \mathbf{H}_i^*\ .$$

The vectors $\mathbf{c}_i$ and $\mathbf{d}_i$ change from $\mathbf{H}_i$ and $\mathbf{H}_i^*$ to $\mathbf{B}_i$ and $\mathbf{B}_i^*$ accordingly:

$$\begin{aligned}\mathbf{c}_i &= (0,\ 0,\ r,\ 0,\ \rho_i,\ x_i,\ 0,\ x_i,\ r')_{\mathbf{H}_i} = (0,\ 0,\ r,\ 0,\ \rho_i,\ \boxed{r'},\ 0,\ x_i,\ r')_{\mathbf{B}_i}\\ \mathbf{d}_i &= (y_i^1, y_i^0, \sigma_i, \pi_i, 0, \Delta y_i, y_i^1, y_i^0, \tau_i)_{\mathbf{H}_i^*} = (y_i^1,\ y_i^0, \sigma_i, \pi_i, 0, \boxed{x_i \Delta y_i / r'}, y_i^1, y_i^0, \tau_i)_{\mathbf{B}_i^*}\end{aligned}$$

while the vectors $\mathbf{c}_{\ell,i}, \mathbf{d}_{k,i}$ have the same writings in $\mathbf{H}_i, \mathbf{H}_i^*$ as in $\mathbf{B}_i, \mathbf{B}_i^*$ because their 6-th coordinates are 0. In summary, in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in G_5^*, else we are in G_6^*. The change is formal.

Game G_7^* : We perform a formal basis change to clean the 6-th coordinate of $\mathbf{c}_i$. The bases are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}_{6,9} \cdot \mathbf{H}_i \qquad\qquad \mathbf{B}_i^* = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}_{6,9} \cdot \mathbf{H}_i^* \ .$$

The vectors $\mathbf{c}_i$ and $\mathbf{d}_i$ change from $\mathbf{H}_i$ and $\mathbf{H}_i^*$ to $\mathbf{B}_i$ and $\mathbf{B}_i^*$ accordingly:

$$\begin{aligned}
\mathbf{c}_i &= (0,\ 0,\ r,\ 0,\ \rho_i,\ r',\ 0,\ x_i,\ r')_{\mathbf{H}_i} = (0,\ 0,\ r,\ 0,\ \rho_i,\ r'-r',\ 0,\ x_i,\ r')_{\mathbf{B}_i} \\
&= (0,\ 0,\ r,\ 0,\ \rho_i,\ \boxed{0},\ 0,\ x_i,\ r')_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ x_i\Delta y_i/r',\ y_i^1,\ y_i^0,\ \tau_i)_{\mathbf{H}_i^*} \\
&= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ x_i\Delta y_i/r',\ y_i^1,\ y_i^0,\ \boxed{\tau_i + x_i\Delta y_i/r'})_{\mathbf{B}_i^*} \ .
\end{aligned}$$

The vectors $\mathbf{c}_{\ell,i}, \mathbf{d}_{k,i}$ have the same writings in $\mathbf{H}_i, \mathbf{H}_i^*$ as in $\mathbf{B}_i, \mathbf{B}_i^*$ because their $(6,9)$-th coordinates are 0. In summary, in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in G_6^*, else we are in G_7^*. The change is formal.

Game G_8^* : We perform a formal basis change to clean the 6-th coordinate of $\mathbf{d}_i$. After G_7^*, the vectors are of the form:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}; \mathbf{c}_i = (0,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0,\ x_i,\ r')_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ x_i\Delta y_i/r',\ y_i^1,\ y_i^0,\ \tau'_i)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0,\ y_{k,i},\ y_{k,i},\ 0)_{\mathbf{B}_i^*} \ .
\end{aligned}$$

We recall that we are in the selective games and $(x_i, y_i^1, y_i^0)_{i=1}^H$ are known in advance. For each i, we define $\alpha_i := -\frac{x_i\Delta y_i}{r'\tau'_i}$, which is well-defined as $r'\tau'_i \neq 0$, and the bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ are changed to $(\mathbf{B}_i, \mathbf{B}_i^*)$ using:

$$\mathbf{B}_i = \begin{bmatrix} 1 & \alpha_i \\ 0 & 1 \end{bmatrix}_{6,9} \cdot \mathbf{H}_i \qquad\qquad \mathbf{B}_i^* = \begin{bmatrix} 1 & 0 \\ -\alpha_i & 1 \end{bmatrix}_{6,9} \cdot \mathbf{H}_i^* \ .$$

It is important that all α_i is defined only once for i, not depending on ℓ nor k, and therefore they can be used for the unique basis changing matrices. The vectors $\mathbf{c}_i$ and $\mathbf{d}_i$ change from $\mathbf{H}_i$ and $\mathbf{H}_i^*$ to $\mathbf{B}_i$ and $\mathbf{B}_i^*$ accordingly:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{H}_i} = (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{c}_i &= (0,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0,\ x_i,\ r')_{\mathbf{H}_i} = (0,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0,\ x_i,\ r')_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ x_i\Delta y_i/r',\ y_i^1,\ y_i^0,\ \tau'_i)_{\mathbf{H}_i^*} \\
&= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ x_i\Delta y_i/r' + \alpha_i\tau'_i,\ y_i^1,\ y_i^0,\ \tau'_i)_{\mathbf{B}_i^*} \\
&= (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ \boxed{0},\ y_i^1,\ y_i^0,\ \tau'_i)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0,\ y_{k,i},\ y_{k,i},\ 0)_{\mathbf{H}_i^*} \\
&= (y_{k,i},\ y_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0,\ y_{k,i},\ y_{k,i},\ 0)_{\mathbf{B}_i^*} \ .
\end{aligned}$$

In bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in G_7^*, else we are in G_8^*. The change is formal and thus $\Pr[\mathsf{G}_8^* = 1] = \Pr[\mathsf{G}_7^* = 1]$.

The above games demonstrate relation (5). We now employ the complexity leveraging argument. Let us fix $j \in \{5,6,7\}$. For $t \in \{j, j+1\}$ let $\mathbf{Adv}_t(\mathcal{A}) \coloneqq |\Pr[\mathsf{G}_t(\mathcal{A}) = 1] - 1/2|$ denote the advantage of a ppt adversary $\mathcal{A}$ in game G_t. We build a ppt adversary $\mathcal{B}^*$ playing against G^*_t such that its advantage $\mathbf{Adv}^*_t(\mathcal{B}^*) \coloneqq |\Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1] - 1/2|$ equals $\gamma \cdot \mathbf{Adv}_t(\mathcal{A})$ for $t \in \{j, j+1\}$, for some constant γ. The adversary $\mathcal{B}^*$ first guesses for the values $(x_i, y_i^0, y_i^1)_{i\in[H]}$, chooses $r' \xleftarrow{\$} \mathbb{Z}_q^*$ and two random secret sharings $(\tau_i)_{i=1}^H$, $(\tau'_i)_{i=1}^H$ of 0, with $\tau_i, \tau'_i \neq 0$ for all i, and define the event E that the guess is correct on the values $(x_i, y_i^0, y_i^1)_{i\in[H]}$ and $\tau'_i - \tau_i = x_i \Delta y_i / r'$, where $\Delta y_i \coloneqq y_i^1 - y_i^0$. When $\mathcal{B}^*$ guesses successfully and $\tau'_i - \tau_i = x_i \Delta y_i / r'$ (call this event E), then the simulation of $\mathcal{A}$'s view in G_t is perfect. Otherwise, $\mathcal{B}^*$ aborts the simulation and outputs a random bit b'. Since E happens with some negligible probability γ and is independent of the view of $\mathcal{A}$, we have:

$$\begin{aligned}
&\mathbf{Adv}^*_t(\mathcal{B}^*) = \left|\Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1] - \tfrac{1}{2}\right| \\
&= \left|\Pr[E] \cdot \Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1 \mid E] + \tfrac{\Pr[\neg E]}{2} - \tfrac{1}{2}\right| \\
&= \left|\gamma \cdot \Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1 \mid E] + \tfrac{1-\gamma-1}{2}\right| \overset{(*)}{=} \gamma \cdot \left|\Pr[\mathsf{G}_t(\mathcal{A}) = 1] - \tfrac{1}{2}\right| = \gamma \cdot \mathbf{Adv}_t(\mathcal{A}).
\end{aligned}$$

where $(*)$ comes from the fact that conditioned on E, $\mathcal{B}$ simulates perfectly G_t for $\mathcal{A}$, therefore $\Pr[\mathsf{G}_t(\mathcal{A}) = 1 \mid E] = \Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1 \mid E]$, then we apply the independence between E and $\mathsf{G}_t(\mathcal{A}) = 1$. This concludes that $\Pr[\mathsf{G}_j = 1] = \Pr[\mathsf{G}_{j+1} = 1]$ for any fixed $j \in \{5,6,7\}$, in particular $\Pr[\mathsf{G}_5 = 1] = \Pr[\mathsf{G}_8 = 1]$. After G_8, the vectors are now of the form:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i}; \mathbf{c}_i = (0,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0,\ x_i,\ r')_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^1, y_i^0, \sigma_i, \pi_i, 0, 0, y_i^1, y_i^0, \tau'_i)_{\mathbf{B}^*_i}; \mathbf{d}_{k,i} = (y_{k,i}, y_{k,i}, \sigma_{k,i}, \pi_{k,i}, 0, 0, y_{k,i}, y_{k,i}, 0)_{\mathbf{B}^*_i}.
\end{aligned}$$

We redo the computational swap to move x_i from coordinate 8 back to coordinate 2 of $\mathbf{c}_i$. The calculation is similar, using basis changing matrices that affect coordinates $(2,5,8)$. Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta \coloneqq c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}^*_i)$ are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & -a \\ 0 & 0 & 1 \end{bmatrix}_{2,5,8} \cdot \mathbf{H}_i; \quad \mathbf{B}^*_i = \begin{bmatrix} 1 & -a & 0 \\ 0 & 1 & 0 \\ 0 & a & 1 \end{bmatrix}_{2,5,8} \cdot \mathbf{H}^*_i \ .$$

All vectors are kept secret. We can compute completely $\mathbf{B}_i$ using $[\![a]\!]_1$ to simulate $\mathbf{c}_{\ell,i}, \mathbf{c}_i$:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ x'_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i} \\
\mathbf{c}_i &= (0,\ 0,\ r,\ 0,\ \rho_i,\ 0,\ 0,\ x_i,\ r')_{\mathbf{B}_i} + (0,\ cx_i,\ 0,\ 0,\ bx_i,\ 0,\ 0, -cx_i,\ 0)_{\mathbf{H}_i} \\
&= (0, \boxed{\delta x_i},\ r,\ 0, \boxed{\rho_i + bx_i},\ 0,\ 0, \boxed{(1-\delta)x_i},\ r')_{\mathbf{B}_i} \ .
\end{aligned}$$

We can also write $\mathbf{d}_i, \mathbf{d}_{k,i}$ can be written in $\mathbf{H}^*_i$ and observed their transformation, for instance $\mathbf{d}_i = (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ a(y_i^0 - y_i^0),\ 0,\ y_i^1,\ y_i^0,\ \tau'_i)_{\mathbf{H}^*_i} = (y_i^1,\ y_i^0,\ \sigma_i,\ \pi_i,\ 0,\ 0,\ y_i^1,\ y_i^0,\ \tau'_i)_{\mathbf{B}^*_i}$ and the same calculation can be done for $\mathbf{d}_{k,i}$. This induces an additive loss $2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$. We redo the transition $\mathsf{G}_0 \to \mathsf{G}_1$ to clean coordinates 9 of $\mathbf{c}_i, \mathbf{d}_i$, which leads to an additive loss $2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$. Then, we redo the transition $\mathsf{G}_1 \to \mathsf{G}_2$ to clean coordinates $(7,8)$ of $\mathbf{d}_{k,i}, \mathbf{d}_i$, which is formal. Finally we arrive at G_9 whose vectors are sampled according to D_1, implying $|\Pr[\mathsf{G}_9] - \Pr[\mathsf{G}_0]| \leq 12 \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$ and the proof is completed. □

B.4 Proof of Theorem 9

Theorem 9. *The DMCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* adaptively one-challenge function-hiding *in the ROM, under* static *corruption, if the* SXDH *assumption holds for* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*. More specifically, let* q_e *denote the number of challenge queries for identical messages, and* q_k *denote the number of functional key queries. Then, for any* ppt *adversary* $\mathcal{A}$ *against* $\mathcal{E}$*, we have the following bound:*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \leq (26q_e + 14q_k + 32) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

Proof. The proof is done via a sequence of hybrid games. The games are depicted in Figure 3.

Game G_0: This is the experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}fh\text{-}0}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$. Because we are in the *one-challenge* setting with *static corruption*, the adversary will declare since $\mathsf{Initialize}$ the challenge ciphertext tag tag^*, the challenge function tag $\mathsf{tag\text{-}f}^*$ as well as the set $\mathcal{C} \subset [n]$ of corrupted clients. We define $\mathcal{H} := [n] \setminus \mathcal{C}$. Knowing $\mathsf{tag}^*, \mathsf{tag\text{-}f}^*$, we index by $\ell \in [q_e]$ the ℓ-th group of ciphertext components queried to $\mathcal{O}\mathsf{Enc}$ for $\mathsf{tag}_\ell \neq \mathsf{tag}^*$. Similarly, we index by $k \in [q_k]$ the k-th group of key components queried to $\mathcal{O}\mathsf{KeyGen}$ for $\mathsf{tag\text{-}f}_k \neq \mathsf{tag\text{-}f}^*$.
There are 2 secret sharings of 0, namely $(\tilde{s}_i)_i$ and $(\tilde{t}_i)_i$, that we generate from $\mathsf{Initialize}$. For the tag $\mathsf{tag\text{-}f}_k$ w.r.t non-challenge functional key queries, we denote $\mathsf{H}_2(\mathsf{tag\text{-}f}_k) \to [\![\mu_k]\!]_2$ and define $s_{k,i} := \mu_k \cdot \tilde{s}_i$. Similarly, for the only challenge functional key query to $\mathsf{DKeyGen}$ corresponding to $\mathsf{tag\text{-}f}^*$, we denote $\mathsf{H}_2(\mathsf{tag\text{-}f}^*) \to [\![\mu]\!]_2$ and define $s_i := \mu \cdot \tilde{s}_i$. We remark that for all $k \in [q_k]$, $(s_{k,i})_i$ is a secret sharing of 0, and the same holds for $(s_i)_i$ as well.
In the same manner, for the ℓ-th non-challenge tag tag, we write $\mathsf{H}_1(\mathsf{tag}_\ell) \to [\![\omega_\ell]\!]_1$ and $t_{\ell,i} := \omega_\ell \cdot \tilde{t}_i$. For the challenge tag tag^*, we denote $\mathsf{H}_1(\mathsf{tag}^*) \to [\![\omega]\!]_1$ and $t_i := \omega \cdot \tilde{t}_i$. In the end, the challenger provides:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ 0,\ \omega_\ell \tilde{t}_i,\ 0,\ \rho_{\ell,i},\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ \mu_k \tilde{s}_i,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu_k,\ \pi_{k,i},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i} \\
\mathbf{c}_i &= (x^0_i,\ \omega,\ 0,\ 0,\ 0,\ 0,\ 0,\ \omega \tilde{t}_i,\ 0,\ \rho_i,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y^0_i,\ \mu \tilde{s}_i,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu,\ \pi_i,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0)_{\mathbf{B}^*_i}
\end{aligned}$$

Note that the admissibility condition in Definition 4 requires that $x^0_i = x^1_i$ (resp. $y^0_i = y^1_i$) for all queries to $\mathcal{O}\mathsf{Enc}$ (resp. $\mathcal{O}\mathsf{DKeyGen}$) where $i \in \mathcal{C}$. Therefore, we are already done for all $i \in \mathcal{C}$. For this reason, all transitions in this prove apply only to pairs of bases $(\mathbf{B}_i, \mathbf{B}^*_i)$ where $i \in \mathcal{H}$. This means in particular that all basis vectors considered in the proof are *hidden* from the adversary.

Game G_1: The vectors are now:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ 0,\ \omega_\ell \tilde{t}_i,\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ \mu_k \tilde{s}_i,\ 0,\ 0,\ 0,\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}^*_i} \\
\mathbf{c}_i &= (x^0_i,\ \omega,\ 0,\ 0,\ 0,\ 0,\ r,\ \omega \tilde{t}_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y^0_i,\ \mu \tilde{s}_i,\ 0,\ 0,\ 0,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}^*_i}
\end{aligned}$$

In this game we replace the shifted secret shares of 0 in $\mathbf{d}_i, \mathbf{d}_{k,i}$ (respectively $\mathbf{c}_i, \mathbf{c}_{\ell,i}$), which are $s_i := \mu \cdot \tilde{s}_i$ and $s_{k,i} := \mu_k \cdot \tilde{s}_i$ (respectively $t_i := \mu \cdot \tilde{t}_i$ and $t_{\ell,i} := \omega_\ell \cdot \tilde{t}_i$), while $\mathsf{H}_1(\mathsf{tag}) \to [\![\omega]\!]_1, \mathsf{H}_1(\mathsf{tag}) \to [\![\omega_\ell]\!]_1, \mathsf{H}_2(\mathsf{tag\text{-}f}) \to [\![\mu]\!]_2, \mathsf{H}_2(\mathsf{tag\text{-}f}) \to [\![\mu_k]\!]_2$ and $\mathsf{H}_1, \mathsf{H}_2$ are modeled as random oracles. We proceed as follows:

$\underline{\mathsf{G}_{0.1}}$**:** We program H_1 at the points $\mathsf{tag}, (\mathsf{tag}_\ell)_{\ell\in[q_e]}$ by sampling $\omega, \omega_\ell \xleftarrow{\$} \mathbb{Z}_q$ and setting $\mathsf{H}_1(\mathsf{tag}) \coloneqq [\![\omega]\!]_1, \mathsf{H}_1(\mathsf{tag\text{-}f}) \coloneqq [\![\omega_\ell]\!]_1$. The same programmation is done for H_2. This gives a perfect simulation and $\Pr[\mathsf{G}_{0.1}] = \Pr[\mathsf{G}_0]$.

$\underline{\mathsf{G}_{0.2}}$**:** We replace the shifted shares in $\mathbf{d}_i, \mathbf{d}_{k,i}$ by random secret shares, while preserving their sum. Our key observation is that: because we are in the *static* corruption model, all *corrupted* i are known since the beginning. More specifically, the secret shares $(\tilde{s}_i)_{i=1}^n$ are generated at setup and $\sum_{i\in\mathcal{H}} \tilde{s}_i = -\left(\sum_{i\in\mathcal{C}} \tilde{s}_i\right)$ is fixed since the beginning. Therefore, upon receiving the challenge tag $\mathsf{tag\text{-}f}$ (that is declared up front by the adversary in the current one-challenge setting) as well as all other non-challenge tags $\mathsf{tag\text{-}f}_k$, thanks to the programmation of the RO from $\mathsf{G}_{0.1}$, all the sums:

$$R \coloneqq \mu \sum_{i\in\mathcal{H}} \tilde{s}_i; \quad R_k \coloneqq \mu_k \sum_{i\in\mathcal{H}} \tilde{s}_i$$

are fixed in advance. We use this observation and the *random-self reducibility* of DDH in $\mathbb{G}_2$ in a sequence of hybrids $\mathsf{G}_{0.1.k}$ over $k \in [q_k+1]\cup\{0\}$ for changing the non-challenge key query $\mathbf{d}_{k,i}$ under $\mathsf{tag\text{-}f}_k$ as well as changing the challenge key query $\mathbf{d}_i$ undef $\mathsf{tag\text{-}f}$.
In the hybrid $\mathsf{G}_{0.1.k}$ with $0 \le k \le q_k$, the first k non-challenge key queries $\mathbf{d}_{k,i}$ are having a random secret shares over $i \in \mathcal{H}$:

$$\mathbf{d}_{k,i} = (y_{k,i},\ \boxed{s_{k,i}},\ 0,\ 0,\ 0,\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*}$$

where $s_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ and $\sum_{i\in\mathcal{H}} s_{k,i} = R_k = \mu_k \cdot \sum_{i\in\mathcal{H}} \tilde{s}_i$. In the hybrid $\mathsf{G}_{0.1.q_k+1}$ we change the challenge key query $\mathbf{d}_i$:

$$\mathbf{d}_i = (y_i^0,\ s_i,\ 0,\ 0,\ 0,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}$$

where $s_i \xleftarrow{\$} \mathbb{Z}_q$ and $\sum_{i\in\mathcal{H}} s_i = R = \mu \cdot \sum_{i\in\mathcal{H}} \tilde{s}_i$. We have $\mathsf{G}_{0.1.0} = \mathsf{G}_{0.1}$ and $\mathsf{G}_{0.1.q_k+1} = \mathsf{G}_{0.2}$.

We describe the transition from $\mathsf{G}_{0.1.k-1}$ to $\mathsf{G}_{0.1.k}$ for $k \in [q_k+1]$, using a DDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ where $c - ab = 0$ or a uniformly random value. Given a ppt adversary $\mathcal{A}$ that can distinguish $\mathsf{G}_{0.1.k-1}$ from $\mathsf{G}_{0.1.k}$ that differ at the k-th key query (being the challenge key if $k = q_k+1$), we build a ppt adversary $\mathcal{B}$ that breaks the DDH:

- The adversary $\mathcal{B}$ uses $[\![a]\!]_2$ to simulate $\mathcal{H}_2(\mathsf{tag\text{-}f}_k)$ (or $\mathcal{H}_2(\mathsf{tag\text{-}f}^*)$ if we are in the last transition to $\mathsf{G}_{0.1.q_k+1}$). This implicitly sets $\mu_k \coloneqq a$.
- The adversary $\mathcal{B}$ samples $\tilde{s}_i \xleftarrow{\$} \mathbb{Z}_q$ for corrupted i, as well as other parameters to output the corrupted keys $(\mathsf{ek}_i, \mathsf{sk}_i)$ to $\mathcal{A}$. Then, $\mathcal{B}$ computes and defines $S_k \coloneqq -\sum_{i\in\mathcal{C}} \tilde{s}_i$.
- Let us denote $H \coloneqq |\mathcal{H}|$ 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 $[\![\mu_k \tilde{s}_i]\!]_2$ for responding to the k-th key query $\mathbf{d}_{k,i}$ (or the challenge $\mathbf{d}_i$ if $k = q_k+1$).
- 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 $b_{k,i} \coloneqq \alpha_{k,i} b + \beta_{k,i}$, $c_{k,i} \coloneqq \alpha_{k,i} c + \beta_{k,i} a$. We note that

$$\begin{cases} [\![b_{k,i}]\!]_2 &= \alpha_{k,i} [\![b]\!]_2 + [\![\beta_{k,i}]\!]_2 \\ [\![c_{k,i}]\!]_2 &= \alpha_{k,i} [\![c]\!]_2 + \beta_{k,i} [\![a]\!]_2 \end{cases}$$

  are efficiently computable from the DDH instance. Then, $\mathcal{B}$ uses $[\![c_{k,i}]\!]_2$ in the simulation of $\mathbf{d}_{k,i}$ (or $\mathbf{d}_i$ in the last hybrid).

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

$$[\![c_{k,H}]\!]_2 := S_k \cdot [\![a]\!]_2 - \sum_{i \in \mathcal{H} \setminus \{H\}} [\![c_{k,i}]\!]_2 \tag{6}$$

where S_k is known in clear from above and other honest $[\![c_{k,i}]\!]_2$ can be computed as explained. The adversary $\mathcal{B}$ then uses $[\![c_{k,H}]\!]_2$ to simulation the H-th key component of the k-th key query. We emphasize that we makes use of the *static* corruption in the simulation for honest i, since we never have to compute 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 $\mathbf{d}_{k,i}[2] = \mu_k \tilde{s}_i := ab_{k,i}$ and we are in $\mathsf{G}_{0.1.k-1}$; Else $\mathbf{d}_{k,i}[2] = s_{k,i} := c_{k,i}$ is a totally uniformly random value such that $\sum_{i \in \mathcal{H}} c_{k,i} + \mu_k \sum_{i \in \mathcal{C}} \tilde{s}_i = aS_k + \mu_k \sum_{i \in \mathcal{C}} \tilde{s}_i = 0$ thanks to (6) and the definition of S_k.

In the end we have $|\Pr[\mathsf{G}_{0.1.k-1} = 1] - \Pr[\mathsf{G}_{0.1.k} = 1]| \leq \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$ and thus $|\Pr[\mathsf{G}_{0.2} = 1] - \Pr[\mathsf{G}_{0.1} = 1]| \leq (q_k + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

$\underline{\mathsf{G}_{0.3}}$**:** We replace the shifted shares $\omega\tilde{t}_i, \omega_\ell \tilde{t}_i$ in $\mathbf{c}_i, \mathbf{c}_{\ell,i}$ by random secret shares $t_i, t_{\ell,i}$ for $i \in \mathcal{H}$, while preserving their sum. The random secret shares $t_i, t_{\ell,i}$ are sampled uniformly at random in $\mathbb{Z}_q$ and satisfy:

$$\sum_{i \in \mathcal{H}} t_i = \omega \sum_{i \in \mathcal{H}} \tilde{t}_i; \quad \sum_{i \in \mathcal{H}} t_{\ell,i} = \omega_\ell \sum_{i \in \mathcal{H}} \tilde{t}_i$$

where $\sum_{i \in \mathcal{H}} \tilde{t}_i$ is fixed from the beginning due to the *static* corruption setting, and the challenge tag is declared up front in the current one-challenge setting. We use the same argument as from $\mathsf{G}_{0.1}$ to $\mathsf{G}_{0.2}$, using DDH in $\mathbb{G}_1$ and with $(q_e + 2)$ hydrids (to change q_e ciphertext queries then the 1 challenge ciphertext). This gives us $|\Pr[\mathsf{G}_{0.3} = 1] - \Pr[\mathsf{G}_{0.2} = 1]| \leq (q_e + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

After arriving at $\mathsf{G}_{0.3}$ the vectors are now having the form:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ 0,\ \boxed{t_{\ell,i}},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ \boxed{s_{k,i}},\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}^*_i} \\
\mathbf{c}_i &= (x^0_i,\ \omega,\ 0,\ 0,\ 0,\ 0,\ 0,\ \boxed{t_i},\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y^0_i,\ \boxed{s_i},\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}^*_i}
\end{aligned}$$

as desired in G_1. As a result $\mathsf{G}_{0.3} = \mathsf{G}_1$ and the total difference in advantages is $|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1]| \leq (q_k + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda) + (q_e + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

Game G_2**:** We perform a computational basis change so as to introduce new random values $r, r_\ell \xleftarrow{\$} \mathbb{Z}_q$ and secret sharings $(\sigma_i)_{i \in \mathcal{H}}$, $(\sigma_{k,i})_{i \in \mathcal{H}}$ of 0, at coordinate 7.

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ \boxed{r_\ell},\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ 0,\ 0,\ 0,\ 0,\ \boxed{\sigma_{k,i}},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}^*_i} \\
\mathbf{c}_i &= (x^0_i,\ \omega,\ 0,\ 0,\ 0,\ 0,\ \boxed{r},\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y^0_i,\ s_i,\ 0,\ 0,\ 0,\ 0,\ \boxed{\sigma_i},\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}^*_i}
\end{aligned}$$

We proceed in two steps:

- $\underline{\mathsf{G}_{1.1}}$: We use the subspace indistinguishability to introduce random values $r, r_1, \ldots, r_{q_e} \xleftarrow{\$} \mathbb{Z}_q$ in the 7-th coordinate of all $\mathbf{c}$-vectors, while keeping $\mathbf{d}_i[7] = \mathbf{d}_{k,i}[7] = 0$:

$$\begin{aligned}\mathbf{c}_i &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ \boxed{r_\ell},\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\\ \mathbf{c}_{\ell,i} &= (x_i^0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ \boxed{r},\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\end{aligned}$$

Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta \coloneqq c - ab$ is either 0 or 1, the basis changing matrices are:

$$\begin{aligned} B_i &\coloneqq \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{2,7} & B_i' &\coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{2,7}\\ \mathbf{B}_i &= B_i \cdot \mathbf{H}_i & \mathbf{B}_i^* &= B_i' \cdot \mathbf{H}_i^* \ . \end{aligned}$$

We compute $\mathbf{B}_i$ and write the $\mathbf{c}$-vectors as follows:

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ 0,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\\ &\quad + (0,\ br_\ell,\ 0,\ 0,\ 0,\ 0,\ cr_\ell,\ 0,\ 0,\ 0,\ 0^4)_{\mathbf{H}_i}\\ &= (x_{\ell,i},\ \boxed{\omega_\ell + br_\ell},\ 0,\ 0,\ 0,\ 0,\ \delta r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\\ \mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ 0,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\\ &\quad + (0,\ br,\ 0,\ 0,\ 0,\ 0,\ cr,\ 0,\ 0,\ 0,\ 0^4)_{\mathbf{H}_i}\\ &= (x_i^0,\ \boxed{\omega + br},\ 0,\ 0,\ 0,\ 0,\ \delta r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\end{aligned}$$

We implicitly update ω to $\omega + br$ and ω_ℓ to $\omega_\ell + br_\ell$. We cannot compute $b_{i,7}^*$ but can write the $\mathbf{d}$-vectors in $\mathcal{H}_i^*$ and observe that they stay invariant in $\mathbf{B}_i^*$ as their 7-th coordinate is 0:

$$\begin{aligned}\mathbf{d}_i &= (y_i^0,\ s_i,\ 0,\ 0,\ 0,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{H}_i^*}\\ &= (y_i^0,\ s_i,\ 0,\ 0,\ 0,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}\\ \mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ 0,\ 0,\ 0,\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{H}_i^*}\\ &= (y_{k,i},\ s_{k,i},\ 0,\ 0,\ 0,\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*}\end{aligned}$$

If $\delta = 0$ we are in G_1 else we are in $\mathsf{G}_{1.1}$. The difference in advantages is $|\Pr[\mathsf{G}_{1.1} = 1] - \Pr[\mathsf{G}_1 = 1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

- $\underline{\mathsf{G}_{1.2}}$: We use DSDH in $\mathbb{G}_2$ to introduce random secret shares of 0, $(\sigma_i)_{i\in\mathcal{H}}$ and $(\sigma_{k,i})_{i\in\mathcal{H}}$ for $k \in [q_k]$, in the 7-th coordinate of the $\mathbf{d}$-vectors. Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta \coloneqq c - ab$ is either 0 or 1, the bases are changed w.r.t the following matrices:

$$\begin{aligned} B_i &\coloneqq \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{2,7} & B_i' &\coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{2,7}\\ \mathbf{B}_i &= B_i \cdot \mathbf{H}_i & \mathbf{B}_i^* &= B_i' \cdot \mathbf{H}_i^* \ . \end{aligned}$$

We compute $\mathbf{B}_i^*$ and write the $\mathbf{d}$-vectors as follows:

$$\begin{aligned}
\mathbf{d}_{k,i} &= (y_{k,i},\ s_i,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
&\quad + (0,\ b\sigma_{k,i},\ 0,\ 0,\ 0,\ 0,\ c\sigma_{k,i},\ 0,\ 0,\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (y_{k,i},\ \boxed{s_{k,i} + b\sigma_{k,i}},\ 0,\ 0,\ 0,\ 0,\ \delta\sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ 0,\ 0,\ 0,\ 0,\ 0,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
&\quad + (0,\ b\sigma_i,\ 0,\ 0,\ 0,\ 0,\ c\sigma_i,\ 0,\ 0,\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (y_i^0,\ \boxed{s_i + b\sigma_i},\ 0,\ 0,\ 0,\ 0,\ \delta\sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

The secret shares $(s_{k,i})_{i\in\mathcal{H}}$ and $(s_i)_{i\in\mathcal{H}}$ are updated to $(s_{k,i} + b\sigma_{k,i})_{i\in\mathcal{H}}$ and $(s_i + b\sigma_i)_{i\in\mathcal{H}}$, by adding a linear shift of $(\sigma_{k,i}, \sigma_i)_{i\in\mathcal{H}}$, and thus stay random independent secret shares of 0. We cannot compute $\mathbf{b}_{i,7}$ but can write the $\mathbf{c}$-vectors in $\mathcal{H}_i$, for $r', r'_\ell \xleftarrow{\$} \mathbb{Z}_q$:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ r'_\ell,\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{H}_i} \\
&= (x_{\ell,i},\ \boxed{r'_\ell + ar_\ell},\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{c}_i &= (x_i^0,\ r',\ 0,\ 0,\ 0,\ 0,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{H}_i} \\
&= (x_i^0,\ \boxed{r' + ar},\ 0,\ 0,\ 0,\ 0,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}
\end{aligned}$$

while simulating $\omega_\ell := r'_\ell + ar_\ell$ and $\omega := r' + ar$. If $\delta = 0$ we are in $\mathsf{G}_{1.1}$ else we are in $\mathsf{G}_{1.2}$. The difference in advantages is $|\Pr[\mathsf{G}_{1.2} = 1] - \Pr[\mathsf{G}_{1.1} = 1]| \le 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

After $\mathsf{G}_{1.2} = \mathsf{G}_2$, the vectors are now:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ \boxed{r_\ell},\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ 0,\ 0,\ 0,\ 0,\ \boxed{\sigma_{k,i}},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ \boxed{r},\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ 0,\ 0,\ 0,\ 0,\ \boxed{\sigma_i},\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

and in total $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1]| \le 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

Game G_3: We perform a sequence of basis changes to alter keys and ciphertexts as follows:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (\boxed{0},\ \boxed{0},\ \boxed{x_{\ell,i}},\ 0,\ \boxed{\omega_\ell},\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (\boxed{0},\ \boxed{0},\ \boxed{y_{k,i}},\ \boxed{y_{k,i}},\ \boxed{s_{k,i}},\ \boxed{s_{k,i}},\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ \boxed{x_i^0},\ 0,\ \boxed{\omega},\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ \boxed{y_i^0},\ 0,\ \boxed{s_i},\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

We detail below the transition from game G_2 to G_3, which is depicted in Figure 5.

- $\underline{\mathsf{G}_{2.0}} = \mathsf{G}_2$.
- $\underline{\mathsf{G}_{2.1}}$: We start with a formal basis change to duplicate the coordinates $(1, 2)$ of $\mathbf{d}_i$ and $\mathbf{d}_{k,i}$ into their coordinates $(3, 5)$, for $i \in \mathcal{H}$ and $k \in [q_k]$:

$$\begin{aligned}
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ \boxed{y_{k,i}},\ 0,\ \boxed{s_{k,i}},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ \boxed{y_i^0},\ 0,\ \boxed{s_i},\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

Game $\mathsf{G}_{2.0} = \mathsf{G}_2$: $\sum_{i=1}^n s_i = \sum_{i=1}^n t_i = \sum_{i=1}^n s_{k,i} = \sum_{i=1}^n t_{\ell,i} = 0$, $\mathsf{H}(\mathsf{tag}_\ell) \to [\![\omega_\ell]\!]_1$, $\mathsf{H}_1(\mathsf{tag}^*) \to [\![\omega]\!]_1$, $\mathsf{H}_2(\mathsf{tag\text{-}f}_k) \to [\![\mu_k]\!]_2$, $\mathsf{H}_2(\mathsf{tag\text{-}f}^*) \to [\![\mu]\!]_2$

$$
\begin{array}{rccc|c|c|c|c|c|c|c|c|c|c}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & 0 & 0 & 0 & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & 0 & 0 & 0 & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^0 & s_i & 0 & 0 & 0 & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{2.1}$: (Formal Duplication)

$$
\begin{array}{rcc|c|c|c|c|c|c|c|c|c}
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & \boxed{y_{k,i}} & 0 & \boxed{s_{k,i}} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{d}_i = (& y_i^0 & s_i & \boxed{y_i^0} & 0 & \boxed{s_i} & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{2.2}$: (Swapping)

$$
\begin{array}{rcc|c|c|c|c|c|c|c|c|c}
\mathbf{c}_{\ell,i} = (& \boxed{0} & \boxed{0} & \boxed{x_{\ell,i}} & 0 & \boxed{\omega_\ell} & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & y_{k,i} & 0 & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{d}_i = (& y_i^0 & s_i & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{2.3}$: (Formal Duplication)

$$
\begin{array}{rcc|c|c|c|c|c|c|c|c|c}
\mathbf{c}_{\ell,i} = (& 0 & 0 & x_{\ell,i} & 0 & \omega_\ell & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & \boxed{x_i^0} & 0 & \boxed{\omega} & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i}
\end{array}
$$

Game $\mathsf{G}_{2.4} = \mathsf{G}_3$: (Swapping)

$$
\begin{array}{rcc|c|c|c|c|c|c|c|c|c}
\mathbf{c}_{\ell,i} = (& 0 & 0 & x_{\ell,i} & 0 & \omega_\ell & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& \boxed{0} & \boxed{0} & y_{k,i} & \boxed{y_{k,i}} & s_{k,i} & \boxed{s_{k,i}} & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^0 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^0 & s_i & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Fig. 5: Transition from G_2 to G_3 in the proof of Theorem 9.

The ciphertexts $\mathbf{c}_i$ and $\mathbf{c}_{\ell,i}$ for $\ell \in [q_e]$ remain unchanged. Let $(\mathbf{H}_i, \mathbf{H}_i^*)$ be a pair of random dual bases. We change the bases w.r.t the following matrices:

$$
B_i := \begin{bmatrix} 1&0&0&0 \\ 0&1&0&0 \\ 1&0&1&0 \\ 0&1&0&1 \end{bmatrix}_{1,2,3,5} \qquad B_i' := \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1&0&-1&0 \\ 0&1&0&-1 \\ 0&0&1&0 \\ 0&0&0&1 \end{bmatrix}_{1,2,3,5}
$$

$$
\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^* \ .
$$

The simulator can write:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i}, 0^4)_{\mathbf{H}_i} \\
&= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ 0,\ 0,\ 0,\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (y_{k,i},\ s_{k,i},\ \boxed{y_{k,i}},\ 0,\ \boxed{s_{k,i}},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{H}_i} \\
&= (x_i^0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ 0,\ 0,\ 0,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (y_i^0,\ s_i,\ \boxed{y_i^0},\ 0,\ \boxed{s_i},\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

Thus, writing ciphertexts and keys in $(\mathbf{H}_i, \mathbf{H}_i^*)$ corresponds to $\mathsf{G}_{2.0}$. Else we are in $\mathsf{G}_{2.1}$. As the basis change is formal, we have $\Pr[\mathsf{G}_{2.1} = 1] = \Pr[\mathsf{G}_{2.0} = 1]$.

- $\underline{\mathsf{G}_{2.2}}$: We perform two computational swaps on the ciphertexts $\mathbf{c}_{\ell,i}$ from $\mathcal{O}\mathsf{Enc}$. We demonstrate how to swap the coordinates $(1, 3)$. Swapping the coordinates $(2, 5)$ can be done similarly. Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta := c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed w.r.t to the following matrices:

$$B_i := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & a & 1 \end{bmatrix}_{1,3,10} \qquad B_i' := \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & -a \\ 0 & 0 & 1 \end{bmatrix}_{1,3,10}$$

$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^* \ .$$

As $\mathbf{B}_i$ can be computed using $[\![a]\!]_1$, the simulator can write:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
&\quad + (-c \cdot x_{\ell,i},\ 0,\ c \cdot x_{\ell,i},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0,\ b \cdot x_{\ell,i},\ 0^4)_{\mathbf{H}_i} \\
&= ((1-\delta) \cdot x_{\ell,i}, \omega_\ell, \delta \cdot x_{\ell,i}, 0, 0, 0, r_\ell, t_{\ell,i}, 0, \rho_{\ell,i} + b \cdot x_{\ell,i}, 0^4)_{\mathbf{B}_i}
\end{aligned}$$

The random value $\rho_{\ell,i}$ is implicitly updated to $\rho_{\ell,i} + bx_{\ell,i}$. The challenge ciphertext $\mathbf{c}_i$ which remains unchanged in this hybrid can be written completely in $\mathbf{B}_i$. The basis vectors $\mathbf{b}_{i,1}^*$ and $\mathbf{b}_{i,3}^*$ cannot be computed due to the lack of $[\![a]\!]_2$. However, the simulator can write keys directly in $\mathbf{H}_i^*$ to observe how they will change:

$$\begin{aligned}
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ y_{k,i},\ 0,\ s_{k,i},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (y_{k,i}, s_{k,i}, y_{k,i}, 0, s_{k,i}, 0, \sigma_{k,i},\ \mu_k, \pi_{k,i}, \boxed{0 + y_{k,i} - y_{k,i}}, 0^4)_{\mathbf{B}_i^*} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (y_i^0,\ s_i,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ \boxed{0 + y_i^0 - y_i^0},\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

If $\delta = 0$ we are not swapping, else we are swapping $x_{\ell,i}$ from the 1-st to the 3-rd coordinate of $\mathbf{c}_{\ell,i}$. In the end, combining the two swappings gives us $|\Pr[\mathsf{G}_{2.2} = 1] - \Pr[\mathsf{G}_{2.1} = 1]| \leq 4 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

- $\underline{\mathsf{G}_{2.3}}$: We continue with the challenge ciphertexts $\mathbf{c}_i$ for $i \in \mathcal{H}$. We duplicate the coordinates $(1,2)$ into coordinates $(4,6)$. Consider the following basis changing matrices:

$$B_i \coloneqq \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{1,2,4,6} \qquad B'_i \coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}_{1,2,4,6}$$
$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}^*_i = B'_i \cdot \mathbf{H}^*_i \ .$$

Then the simulator can write:

$$\begin{aligned}
\mathbf{c}_{\ell,i} &= (0,\ 0,\ x_{\ell,i},\ 0,\ \omega_\ell,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i}, 0^4)_{\mathbf{H}_i} \\
&= (0,\ 0,\ x_{\ell,i},\ 0,\ \omega_\ell,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ y_{k,i},\ 0,\ s_{k,i},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{H}^*_i} \\
&= (y_{k,i},\ s_{k,i},\ y_{k,i},\ 0,\ s_{k,i},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}^*_i} \\
\mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ 0,\ 0,\ 0,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{H}_i} \\
&= (x_i^0,\ \omega,\ 0,\ \boxed{x_i^0},\ 0,\ \boxed{\omega},\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^0,\ s_i,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{H}^*_i} \\
&= (y_i^0,\ s_i,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}^*_i}
\end{aligned}$$

Thus, writing ciphertexts and keys in $(\mathbf{H}_i, \mathbf{H}^*_i)$ corresponds to $\mathsf{G}_{2.2}$, else we are in $\mathsf{G}_{2.3}$. As the basis change is formal, we have $\Pr[\mathsf{G}_{2.3} = 1] = \Pr[\mathsf{G}_{2.2} = 1]$.

- $\underline{\mathsf{G}_{2.4} = \mathsf{G}_3}$: Finally, we perform two computational swaps on the non-challenge functional keys $\mathbf{d}_{k,i}$. We demonstrate how to swap coordinates $(1,4)$. Coordinates $(2,6)$ can be swapped similarly. Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta \coloneqq c - ab$ is either 0 or 1, we define:

$$B_i \coloneqq \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & -a \\ 0 & 0 & 1 \end{bmatrix}_{1,4,9} \qquad B'_i \coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -a & a & 1 \end{bmatrix}_{1,4,9}$$
$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}^*_i = B'_i \cdot \mathbf{H}^*_i \ .$$

The basis vector $\mathbf{b}^*_{i,9}$ can be computed using $[\![a]\!]_2$. Then the simulator can write:

$$\begin{aligned}
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ y_{k,i},\ 0,\ s_{k,i},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}^*_i} \\
&\quad + (-c \cdot y_{k,i},\ 0,\ 0,\ c \cdot y_{k,i},\ 0,\ 0,\ 0,\ 0,\ b \cdot y_{k,i},\ 0,\ 0^4)_{\mathbf{H}^*_i} \\
&= ((1-\delta)y_{k,i}, s_{k,i}, y_{k,i}, \delta y_{k,i}, s_{k,i}, 0, \sigma_{k,i}, \mu_k, \pi_{k,i} + b \cdot y_{k,i}, 0, 0^4)_{\mathbf{B}^*_i}
\end{aligned}$$

The random value $\pi_{k,i}$ is implicitly updated to $\pi_{k,i} + by_{k,i}$. The challenge secret key $\mathbf{d}_i$ can be written completely in $\mathbf{B}^*_i$. We cannot compute $\mathbf{b}_{i,1}$ and $\mathbf{b}_{i,4}$ due to the lack of $[\![a]\!]_1$. However, the simulator can write $\mathbf{c}_i$ and $\mathbf{c}_{\ell,i}$ directly in $\mathbf{H}_i$ since their representation is invariant under

the above basis change:

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (0,\ 0,\ x_{\ell,i},\ 0,\ \omega_\ell,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i}, 0^4_{\mathbf{H}_i}\\ &= (0,\ 0,\ x_{\ell,i},\ 0,\ \omega_\ell,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\\ \mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{H}_i}\\ &= (x_i^0,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ \boxed{0 - ax_i^0 + ax_i^0},\ \rho_i,\ 0^4)_{\mathbf{B}_i}\end{aligned}$$

If $\delta = 0$, then we are not swapping, else we are swapping the coordinates $(1,4)$ of $\mathbf{d}_{k,i}$. After two swappings, we obtain $|\Pr[\mathsf{G}_{2.4} = 1] - \Pr[\mathsf{G}_{2.3} = 1]| \leq 4 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

In the end, we have

$$|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq 4 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda) + 4 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)\ .$$

Game G_4: We flip the bit in the first coordinates. More specifically, we transform $(\mathbf{c}_i, \mathbf{d}_i)$ into the following form:

$$\begin{aligned}\mathbf{c}_i &= (\boxed{x_i^1},\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\\ \mathbf{d}_i &= (\boxed{y_i^1},\ \boxed{s_i'},\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}\ ,\end{aligned}$$

where $z_i = x_i^1 y_i^1 - x_i^0 y_i^0$ and $s_i' = s_i - z_i/\omega$. As we are in the one-challenge setting, both challenge tags, specifically tag for ciphertext and $\mathsf{tag\text{-}f}$ for functional key, are announced by the adversary during $\mathsf{Initialize}$ thus we do not have to guess ω nor μ.

As in [24], we obtain adaptive security by employing the complexity leveraging technique: First, we prove perfect security in the *fully selective* variant of the involved games, where the adversary needs to announce both challenge messages and challenge functions before the setup. Then, using a guessing argument, we obtain the same security guarantees in the adaptive games. The guessing incurs an exponential security loss. However, as the security in the selective game is *perfect*, i.e. the advantage of the adversary is exactly 0, the security loss is multiplied by a zero term. So the overall adaptive security is preserved.

For the transition $\mathsf{G}_3 \to \mathsf{G}_4$ we guess in advance $(x_i^1, x_i^0, y_i^0, y_i^1)_{i\in\mathcal{H}}$, program the RO of $\mathcal{H}_1$ to output $[\![\omega]\!]_1$ for $\omega \xleftarrow{\$} \mathbb{Z}_q$, on the challenge ciphertext tag tag^*, sample $\omega_i', \omega_i'' \xleftarrow{\$} \mathbb{Z}_q^*$ for all $i \in \mathcal{H}$. We define an event E to say that the guesses are correct when $\omega_i' = \omega y_i^0, \omega_i'' = \omega y_i^1$.

Step 1. For $j \in \{3,4\}$ we denote by G_j^* the fully selective variant of G_j, where E is assumed true. Using a formal basis change, we show that the advantage of any ppt adversary in distinguishing between G_3^* and G_4^* is exactly 0. We remark that assuming E true implies $\omega \neq 0, y_i^0 \neq 0, y_i^1 \neq 0$. Let $(\mathbf{H}_i, \mathbf{H}_i^*)$ be a pair of random dual bases and $z_i \coloneqq x_i^1 y_i^1 - x_i^0 y_i^0$. The basis change is done w.r.t the following matrices:

$$B_i \coloneqq \begin{bmatrix} y_i^1/y_i^0 & 0 \\ -z_i/\omega_i' & 1 \end{bmatrix}_{1,2} \qquad B_i' \coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} y_i^0/y_i^1 & z_i/\omega_i'' \\ 0 & 1 \end{bmatrix}_{1,2}$$

$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^*\ .$$

The simulator can write:

$$\begin{aligned}\mathbf{c}_i &= (x_i^0,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{H}_i}\\ &= (x_i^1,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\\ \mathbf{d}_i &= (y_i^0,\ s_i,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{H}_i^*}\\ &= (y_i^1,\ s_i - z_i/\omega,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}\end{aligned}$$

Furthermore, we observe that ciphertexts $\mathbf{c}_{\ell,i}$ and keys $\mathbf{d}_{k,i}$ are invariant under this basis change because their first two coordinates are 0. Thus, writing $(\mathbf{c}_i, \mathbf{d}_i)$ in $(\mathbf{H}_i, \mathbf{H}_i^*)$ corresponds to G_4^*. Else we are in G_3^*. Since the basis change is formal, we have $\Pr[\mathsf{G}_4^* = 1] = \Pr[\mathsf{G}_3^* = 1]$.

Step 2. For $j \in \{3, 4\}$ let $\mathbf{Adv}_j(\mathcal{A}) \coloneqq |\Pr[\mathsf{G}_j(\mathcal{A}) = 1] - 1/2|$ denote the advantage of a ppt adversary $\mathcal{A}$ in game G_j. We build a ppt adversary $\mathcal{B}^*$ playing against $\mathbb{G}_j^*$ such that its advantage $\mathbf{Adv}_j^*(\mathcal{B}^*) \coloneqq |\Pr[\mathsf{G}_j^*(\mathcal{B}^*) = 1] - 1/2|$ equals $\gamma \cdot \mathbf{Adv}_j(\mathcal{A})$ for $j \in \{3, 4\}$ and $\Pr[E] = \gamma$.

$\mathcal{B}^*$ first guesses $\mathbf{x}_i, \mathbf{y}_i \xleftarrow{\$} \mathbb{Z}_q^2 \cup \{\perp\}$ for all $i \in [n]$ which it sends to its fully selective game G_j^*. That is, each guess $\mathbf{x}_i$ is either a pair of values (x_i^0, x_i^1) queried to $\mathcal{O}\mathsf{Enc}$, or $\perp$ which means no query to $\mathcal{O}\mathsf{Enc}$. Similarly, $\mathbf{y}_i$ is either interpreted as a query (y_i^0, y_i^1) to $\mathcal{O}\mathsf{DKeyGen}$ or no query to $\mathcal{O}\mathsf{DKeyGen}$. Then, adversary $\mathcal{B}$ programs the RO of $\mathcal{H}_1$ to output $[\![\omega]\!]_1$ for $\omega \xleftarrow{\$} \mathbb{Z}_q$, on the challenge tag tag^*, samples $\omega_i', \omega_i'' \xleftarrow{\$} \mathbb{Z}_q^*$ for all $i \in \mathcal{H}$, and simulates the view of $\mathcal{A}$ using its own oracles. When $\omega_i' = \omega y_i^0, \omega_i'' = \omega y_i^1$ (call this event E), then the simulation of $\mathcal{A}$'s view in G_j is perfect. Otherwise, $\mathcal{B}^*$ aborts the simulation and outputs a random bit b'. Since E happens with probability γ and is independent of the view of $\mathcal{A}$, we have:

$$\begin{aligned}
\mathbf{Adv}_j^*(\mathcal{B}^*) &= \left|\Pr[\mathsf{G}_j^*(\mathcal{B}^*) = 1] - \frac{1}{2}\right| \\
&= \left|\Pr[E] \cdot \Pr[\mathsf{G}_j^*(\mathcal{B}^*) = 1 \mid E] + \frac{\Pr[\neg E]}{2} - \frac{1}{2}\right| \\
&= \left|\gamma \cdot \Pr[\mathsf{G}_j^*(\mathcal{B}^*) = 1 \mid E] + \frac{1 - \gamma - 1}{2}\right| \\
&\overset{(*)}{=} \gamma \cdot \left|\Pr[\mathsf{G}_j(\mathcal{A}) = 1] - \frac{1}{2}\right| \\
&= \gamma \cdot \mathbf{Adv}_j(\mathcal{A})
\end{aligned}$$

where $(*)$ comes from the fact that conditioned on E, $\mathcal{B}$ simulates perfectly G_j for $\mathcal{A}$, therefore $\Pr[\mathsf{G}_j(\mathcal{A}) = 1 \mid E] = \Pr[\mathsf{G}_j^*(\mathcal{B}^*) = 1 \mid E]$, then we apply the independence between E and $[\mathsf{G}_j(\mathcal{A}) = 1]$. Combining the above argument with $\Pr[\mathsf{G}_4^* = 1] = \Pr[\mathsf{G}_3^* = 1]$ implies $\Pr[\mathsf{G}_4 = 1] = \Pr[\mathsf{G}_3 = 1]$.

Game G_5: We perform a computational swapping on the coordinates $(2, 6)$ of the non-challenge functional keys $\mathbf{d}_{k,i}$. The challenge functional key $\mathbf{d}_i$ and the $\mathbf{c}$-vectors remain unchanged.

$$\mathbf{d}_{k,i} = (0,\ \boxed{s_{k,i}},\ y_{k,i},\ y_{k,i},\ s_{k,i},\ \boxed{0},\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*}$$

Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta \coloneqq c - ab$ is either 0 or 1, we define:

$$B_i \coloneqq \begin{bmatrix} 1 & 0 & -a \\ 0 & 1 & a \\ 0 & 0 & 1 \end{bmatrix}_{2,6,9} \qquad B_i' \coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ a & -a & 1 \end{bmatrix}_{2,6,9}$$

$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^* \ .$$

The basis vector $\mathbf{b}_{i,9}^*$ can be computed using $[\![a]\!]_2$. Then the simulator can write:

$$\begin{aligned}
\mathbf{d}_{k,i} &= (0,\ 0,\ y_{k,i},\ y_{k,i},\ s_{k,i},\ s_{k,i},\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
&\quad + (0,\ c \cdot s_{k,i},\ 0,\ 0,\ 0,\ -c \cdot s_{k,i},\ 0,\ 0,\ b \cdot s_{k,i},\ 0,\ 0^4)_{\mathbf{H}_i^*} \\
&= (0, \delta \cdot s_{k,i}, y_{k,i}, y_{k,i}, s_{k,i}, (1-\delta)s_{k,i}, \sigma_{k,i}, \mu_k, \pi_{k,i} + b \cdot s_{k,i}, 0, 0^4)_{\mathbf{B}_i^*}
\end{aligned}$$

Game $\mathsf{G}_{5.0} = \mathsf{G}_5$: $\sum_{i=1}^n s_i = \sum_{i=1}^n t_i = \sum_{i=1}^n s_{k,i} = \sum_{i=1}^n t_{\ell,i} = 0$, $\mathsf{H}(\mathsf{tag}_\ell) \to [\![\omega_\ell]\!]_1$, $\mathsf{H}_1(\mathsf{tag}^*) \to [\![\omega]\!]_1$, $\mathsf{H}_2(\mathsf{tag\text{-}f}_k) \to [\![\mu_k]\!]_2$, $\mathsf{H}_2(\mathsf{tag\text{-}f}^*) \to [\![\mu]\!]_2$

$$
\begin{array}{rccc|ccc|ccc|ccc|c}
\mathbf{c}_{\ell,i} = (& 0 & 0 & x_{\ell,i} & 0 & \omega_\ell & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& 0 & s_{k,i} & y_{k,i} & y_{k,i} & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s'_i & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{5.1}$: (Hybrids of Secret-Sharing Swapping (Lemma 8) for each $\mathbf{c}_{\ell,i}$)

$$
\begin{array}{rccc|ccc|ccc|ccc|c}
\mathbf{c}_{\ell,i} = (& 0 & \boxed{\omega_\ell} & x_{\ell,i} & 0 & \boxed{0} & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& 0 & s_{k,i} & y_{k,i} & y_{k,i} & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s'_i & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{5.2}$: (Hybrids of Secret-Sharing Swapping (Lemma 8) for each $\mathbf{d}_{k,i}$)

$$
\begin{array}{rccc|ccc|ccc|ccc|c}
\mathbf{c}_{\ell,i} = (& 0 & \omega_\ell & x_{\ell,i} & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& \boxed{y_{k,i}} & s_{k,i} & y_{k,i} & \boxed{0} & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s'_i & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{5.3} = \mathsf{G}_6$: (Hybrids of Secret-Sharing Swapping (Lemma 8) for each $\mathbf{c}_{\ell,i}$)

$$
\begin{array}{rccc|ccc|ccc|ccc|c}
\mathbf{c}_{\ell,i} = (& \boxed{x_{\ell,i}} & \omega_\ell & \boxed{0} & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & y_{k,i} & 0 & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s'_i & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Fig. 6: Transition from G_5 to G_6 in Theorem 9.

The random value $\pi_{k,i}$ is implicitly updated to $\pi_{k,i} + bs_{k,i}$. The challenge secret key $\mathbf{d}_i$ can be written completely in $\mathbf{B}_i^*$. We cannot compute $\mathbf{b}_{i,2}$ and $\mathbf{b}_{i,6}$ due to the lack of $[\![a]\!]_1$. However, the simulator can write $\mathbf{c}_i$ and $\mathbf{c}_{\ell,i}$ directly in $\mathbf{H}_i$ since their representation is invariant under the above basis change. If $\delta = 0$, then we are in G_4, else we are G_5. Finally, we obtain $|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

Game G_6**:** We perform a sequence of basis changes to alter keys and ciphertexts as follows:

$$
\begin{aligned}
\mathbf{c}_{\ell,i} &= (\boxed{x_{\ell,i}},\ \boxed{\omega_\ell},\ \boxed{0},\ 0,\ \boxed{0},\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} &= (\boxed{y_{k,i}},\ s_{k,i},\ y_{k,i},\ \boxed{0},\ s_{k,i},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{c}_i &= (x_i^1,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\
\mathbf{d}_i &= (y_i^1,\ s'_i,\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}
$$

We detail below the transition from game G_5 to G_6, which is depicted in Figure 6. The *secret-sharing swapping* lemma (Lemma 8) is applied ubiquitously in this transition, we refer to the overview of the proof in Section 6.1 for high-level ideas.

- $\mathsf{G}_{5.0} = \mathsf{G}_5$.

- $\underline{\mathsf{G}_{5.1}}$: We apply Lemma 8 in hybrids on $\ell \in [q_e]$, moving from one to the next for swapping ω_ℓ back to coordinate 2 in $\mathbf{c}_{\ell,i}$ for $i \in \mathcal{H}$. The 9 coordinates affected, in the order w.r.t the statement of Lemma 8 so that they form a subspace of dimension 9, are $(2, 5, 7, 9, 10, 11, 12, 13, 14)$. In the ℓ-th hybrid, we apply the lemma while keeping the same variable names for all $\mathbf{c}$-vectors and $\mathbf{d}$-vectors as in the proof of Theorem 9, except that we swap the names of $\mathbf{c}_i$ and $\mathbf{c}_{\ell,i}$. Coordinates $(2, 5)$ of $\mathbf{d}_i$ contain the values (s'_i, s_i). For a correct application of the lemma, we need to check that $\sum_{i\in\mathcal{H}} s'_i = \sum_{i\in\mathcal{H}} s_i$. We have

$$\begin{aligned}\sum_{i\in\mathcal{H}} s'_i &= \sum_{i\in\mathcal{H}} \left(s_i - \frac{x_i^1 y_i^1 - x_i^0 y_i^0}{\omega}\right)\\ &= \left(\sum_{i\in\mathcal{H}} s_i\right) - \frac{1}{\omega}\left(\sum_{i\in\mathcal{H}} x_i^1 y_i^1\right) + \frac{1}{\omega}\left(\sum_{i\in\mathcal{H}} x_i^0 y_i^0\right)\\ &= \sum_{i\in\mathcal{H}} s_i\end{aligned}$$

  and we use the fact that $\sum_{i\in\mathcal{H}} x_i^1 y_i^1 = \sum_{i\in\mathcal{H}} x_i^0 y_i^0$ from the admissibility condition of Definition 4.

  In the end $|\Pr[\mathsf{G}_{5.1} = 1] - \Pr[\mathsf{G}_{5.0} = 1]| \leq 12q_e \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.
- $\underline{\mathsf{G}_{5.2}}$: We apply Lemma 8 in hybrids on $k \in [q_k]$, moving from one to the next for swapping $y_{k,i}$ back to coordinate 1 in $\mathbf{d}_{k,i}$. The 9 coordinates affected, in the order w.r.t the statement of Lemma 8 so that they form a subspace of dimension 9, are $(1, 4, 8, 10, 9, 11, 12, 13, 14)$.

  We remark that the lemma is applied on the dual bases, *i.e.* $\mathbb{G}_1$ and $\mathbb{G}_2$ are swapped. Hence, we must check that $\sum_{i\in\mathcal{H}} y_{k,i} x_i^0 = \sum_{i\in\mathcal{H}} y_{k,i} x_i^1$. Again, this is true thanks to the admissibility condition of Definition 4. Moreover, the family of vectors $(\mathbf{c}_{\ell,i})_{\ell,i}$ satisfy the requirement in Lemma 8 that coordinates $(1, 3)$ in these vectors are the same (they are 0). In the end $|\Pr[\mathsf{G}_{5.2} = 1] - \Pr[\mathsf{G}_{5.1} = 1]| \leq 12q_k \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.
- $\underline{\mathsf{G}_{5.3}} = \mathsf{G}_6$: We apply Lemma 8 in hybrids on $\ell \in [q_e]$, moving from one to the next for swapping $x_{\ell,i}$ back to coordinate 1 in $\mathbf{c}_{\ell,i}$. The 9 coordinates affected, in the order w.r.t the statement of Lemma 8 so that they form a subspace of dimension 9, are $(1, 3, 7, 9, 10, 11, 12, 13, 14)$. For a correct application, we must check that $\sum_{i\in\mathcal{H}} x_{\ell,i} y_i^0 = \sum_{i\in\mathcal{H}} x_{\ell,i} y_i^1$ which is satisfied thanks to the admissibility of Definition 4.

  Finally, $|\Pr[\mathsf{G}_{5.3} = 1] - \Pr[\mathsf{G}_{5.2} = 1]| \leq 12q_e \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

In the end, we have $|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq (24q_e + 12q_k) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

Game G_7: We perform a computational basis change to clean coordinates $(3, 4, 5, 6, 7)$.

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ \boxed{0},\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\\ \mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ \boxed{0},\ 0,\ \boxed{0},\ 0,\ \boxed{0},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*}\\ \mathbf{c}_i &= (x_i^1,\ \omega,\ 0,\ \boxed{0},\ 0,\ \boxed{0},\ \boxed{0},\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\\ \mathbf{d}_i &= (y_i^1,\ s'_i,\ \boxed{0},\ 0,\ \boxed{0},\ 0,\ \boxed{0},\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}\end{aligned}$$

The subsequence of Games from G_6 to G_7 is depicted in Figure 7. We detail the transitions below.

- $\underline{\mathsf{G}_{6.0}} = \mathsf{G}_6$.

Game $\mathsf{G}_{6.0} = \mathsf{G}_6$:

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & y_{k,i} & 0 & s_{k,i} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s_i' & y_i^0 & 0 & s_i & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{6.1}$: (Formal Step)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & \boxed{0} & 0 & \boxed{0} & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s_i' & \boxed{y_i^0 - y_i^1} & 0 & \boxed{s_i - s_i'} & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{6.2}$: (Subspace)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & 0 & 0 & 0 & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & x_i^0 & 0 & \omega & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s_i' & \boxed{0} & 0 & \boxed{0} & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game $\mathsf{G}_{6.3}$: (Subspace)

$$
\begin{array}{rcccccccccccc}
\mathbf{c}_{\ell,i} = (& x_{\ell,i} & \omega_\ell & 0 & 0 & 0 & 0 & r_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i} = (& y_{k,i} & s_{k,i} & 0 & 0 & 0 & 0 & \sigma_{k,i} & \mu_k & \pi_{k,i} & 0 & 0^4 &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i = (& x_i^1 & \omega & 0 & \boxed{0} & 0 & \boxed{0} & r & t_i & 0 & \rho_i & 0^4 &)_{\mathbf{B}_i} \\
\mathbf{d}_i = (& y_i^1 & s_i' & 0 & 0 & 0 & 0 & \sigma_i & \mu & \pi_i & 0 & 0^4 &)_{\mathbf{B}_i^*}
\end{array}
$$

Fig. 7: Transition from G_6 to G_7 in Theorem 9.

- $\mathsf{G}_{6.1}$: We start with a formal basis change to alter the d-vectors as follows:

$$
\begin{aligned}
\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ \boxed{0},\ 0,\ \boxed{0},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*} \\
\mathbf{d}_i &= (y_i^1,\ s_i',\ \boxed{y_i^0 - y_i^1},\ 0,\ \boxed{s_i - s_i'},\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}
\end{aligned}
$$

The c-vectors remain unchanged. Let $(\mathbf{H}_i, \mathbf{H}_i^*)$ be a pair of random dual bases. We change the bases w.r.t the following matrices:

$$
B_i \coloneqq \begin{bmatrix} 1 & 0 & 0\,0 \\ 0 & 1 & 0\,0 \\ -1 & 0 & 1\,0 \\ 0 & -1 & 0\,1 \end{bmatrix}_{1,2,3,5} \qquad B_i' \coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \\ 0\,0\,1\,0 \\ 0\,0\,0\,1 \end{bmatrix}_{1,2,3,5}
$$

$$
\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^* \ .
$$

The simulator can write:

$$\begin{aligned}\mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ y_{k,i},\ 0,\ s_{k,i},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{H}_i^*}\\ &= (y_{k,i},\ s_{k,i},\ \boxed{0},\ 0,\ \boxed{0},\ 0,\ \sigma_{k,i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}_i^*}\\ \mathbf{d}_i &= (y_i^1,\ s_i',\ y_i^0,\ 0,\ s_i,\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{H}_i^*}\\ &= (y_i^1,\ s_i',\ \boxed{y_i^0 - y_i^1},\ 0,\ \boxed{s_i - s_i'},\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}\end{aligned}$$

The c-vectors are invariant under this basis change as their coordinates $(3,5)$ are 0. Thus, writing ciphertexts and keys in $(\mathbf{H}_i, \mathbf{H}_i^*)$ corresponds to $\mathsf{G}_{6.0}$. Else we are in $\mathsf{G}_{6.1}$. As the basis change is formal, we have $\Pr[\mathsf{G}_{6.1} = 1] = \Pr[\mathsf{G}_{6.0} = 1]$.

- $\underline{\mathsf{G}_{6.2}}$: We clean coordinates $(3,5)$ of $\mathbf{d}_i$ using DSDH in $\mathbb{G}_2$.

$$\mathbf{d}_i = (y_i^1,\ s_i',\ \boxed{0},\ 0,\ \boxed{0},\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}$$

We demonstrate the cleaning of coordinate 1. Coordinate 5 can be done similarly. Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta \coloneqq c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed w.r.t to the following matrices:

$$\begin{aligned} B_i &\coloneqq \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{3,9} & B_i' &\coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{3,9}\\ \mathbf{B}_i &= B_i \cdot \mathbf{H}_i & \mathbf{B}_i^* &= B_i' \cdot \mathbf{H}_i^* \ .\end{aligned}$$

As $\mathbf{B}_i^*$ can be computed using $[\![a]\!]_2$, the simulator can write:

$$\begin{aligned}\mathbf{d}_i &= (y_i^1,\ s_i',\ y_i^0 - y_i^1,\ 0,\ s_i - s_i',\ 0,\ \sigma_i,\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}_i^*}\\ &\quad + (0,\ 0,\ -c(y_i^0 - y_i^1),\ 0,\ 0,\ 0,\ 0,\ 0,\ b(y_i^0 - y_i^1),\ 0,\ 0^4)_{\mathbf{H}_i^*}\\ &= (y_i^1, s_i', (1-\delta)(y_i^0 - y_i^1), 0, s_i - s_i', 0, \sigma_i, \mu, \pi_i + b(y_i^0 - y_i^1), 0, 0^4)_{\mathbf{B}_i^*}\end{aligned}$$

The random value π_i is implicitly updated to $\pi_i + b(y_i^0 - y_i^1)$. The remaining keys $\mathbf{d}_{k,i}$ which are not changed in this hybrid can be written completely in $\mathbf{B}_i^*$. The basis vector $\mathbf{b}_{i,3}$ cannot be computed due to the lack of $[\![a]\!]_1$. However, the simulator can write $\mathbf{c}$-vectors directly in $\mathbf{H}_i$, as they are invariant under this basis change because their 3-rd coordinate is 0:

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{H}_i}\\ &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ r_\ell,\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i}\\ \mathbf{c}_i &= (x_i^1,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{H}_i}\\ &= (x_i^1,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}\end{aligned}$$

If $\delta = 1$ we are cleaning the 3-rd coordinate of $\mathbf{d}_i$, else we change nothing. In the end, combining the two cleanings gives us $|\Pr[\mathsf{G}_{6.2} = 1] - \Pr[\mathsf{G}_{6.1} = 1]| \leq 4 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

- $\underline{\mathsf{G}_{6.3}}$: We employ a second subspace indistinguishability to clean coordinates $(4,6)$ of $\mathbf{c}_i$:

$$\mathbf{c}_i = (x_i^1,\ \omega,\ 0,\ \boxed{0},\ 0,\ \boxed{0},\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i}$$

The remaining ciphertexts and keys are not changed. We demonstrate how to clean coordinate 4. Coordinate 6 can be done similarly. Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta \coloneqq c - ab$ is either 0 or 1, we define:

$$B_i \coloneqq \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{4,10} \qquad B'_i \coloneqq \left(B_i^{-1}\right)^\top = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{4,10}$$
$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i \qquad \mathbf{B}^*_i = B'_i \cdot \mathbf{H}^*_i \ .$$

The basis vector $\mathbf{b}_{i,4}$ can be computed using $[\![a]\!]_1$. Then the simulator can write:

$$\begin{aligned}\mathbf{c}_i &= (x_i^1,\ \omega,\ 0,\ x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\ &\quad + (0,\ 0,\ 0,\ -cx_i^0,\ 0,\ 0,\ 0,\ 0,\ 0,\ bx_i^0,\ 0^4)_{\mathbf{H}_i} \\ &= (x_i^1,\ \omega,\ 0,\ (1-\delta)x_i^0,\ 0,\ \omega,\ r,\ t_i,\ 0,\ \rho_i + bx_i^0,\ 0^4)_{\mathbf{B}_i}\end{aligned}$$

The random value ρ_i is implicitly updated to $\rho_i + bx_i^0$. The remaining ciphertexts $\mathbf{c}_{\ell,i}$ which are not changed in this hybrid can be written completely in $\mathbf{B}_i$. The basis vector $\mathbf{b}^*_{i,4}$ cannot be computed due to the lack of $[\![a]\!]_2$. However, $\mathbf{d}$-vectors are invariant under this basis change because their 3-rd coordinate is 0. So the simulator can write them directly in $\mathbf{H}^*_i$.
If $\delta = 0$, then we do nothing, else we are cleaning the 4-th coordinate of $\mathbf{c}_i$. After two cleanings, we obtain $|\Pr[\mathsf{G}_{6.3} = 1] - \Pr[\mathsf{G}_{6.2} = 1]| \le 4 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

- $\underline{\mathsf{G}_{6.4}}$: We clean coordinate 7 of all vectors.

$$\begin{aligned}\mathbf{c}_{\ell,i} &= (x_{\ell,i},\ \omega_\ell,\ 0,\ 0,\ 0,\ 0,\ \boxed{0},\ t_{\ell,i},\ 0,\ \rho_{\ell,i},\ 0^4)_{\mathbf{B}_i} \\ \mathbf{d}_{k,i} &= (y_{k,i},\ s_{k,i},\ 0,\ 0,\ 0,\ 0,\ \boxed{0},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^4)_{\mathbf{B}^*_i} \\ \mathbf{c}_i &= (x_i^1,\ \omega,\ 0,\ 0,\ 0,\ 0,\ \boxed{0},\ t_i,\ 0,\ \rho_i,\ 0^4)_{\mathbf{B}_i} \\ \mathbf{d}_i &= (y_i^1,\ s'_i,\ 0,\ 0,\ 0,\ 0,\ \boxed{0},\ \mu,\ \pi_i,\ 0,\ 0^4)_{\mathbf{B}^*_i}\end{aligned}$$

The transition from $\mathsf{G}_{6.3}$ to $\mathsf{G}_{6.4}$ is inverse to that from G_1 to G_2. Thus, we obtain $|\Pr[\mathsf{G}_{6.4} = 1] - \Pr[\mathsf{G}_{6.3} = 1]| \le 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

In the end, we have

$$|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_6 = 1]| \le 6 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda) + 6 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda) \ .$$

Game G_8: We redo the transition $\mathsf{G}_1 \to \mathsf{G}_2$ to switch back the random independent secret shares $(s_i, s_{k,i})_i$ and $(t_i, t_{\ell,i})$ to shifted secret shares $(\mu\tilde{s}_i, \mu_k\tilde{s}_i)_i$ and $(\omega\tilde{t}_i, \omega_\ell\tilde{t}_i)$, respectively. This incurs an additive loss $|\Pr[\mathsf{G}_8 = 1] - \Pr[\mathsf{G}_7 = 1]| \le (q_e + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda) + (q_k + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

The game G_8 is $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}fh\text{-}1}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$. The difference in advantages is

$$\begin{aligned}|\Pr[\mathsf{G}_8 = 1] - \Pr[\mathsf{G}_0 = 1]| &\le \sum_{i=1}^{8} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i-1} = 1]| \\ &\le (26q_e + 14q_k + 32) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)\end{aligned}$$

and the proof is completed. □

B.5 Security Theorems for Section 6.3

The following theorem states the *weakly function-hiding* security against *incomplete adaptive one-challenge* under *static* corruption of the scheme $\mathcal{E}'$ obtained from the transformation of Section 6.3.

Theorem 12. *Let* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a DMCFE scheme that is* weakly function-hiding *against* complete adaptive one-challenge *under* static *corruption following Definition 4. Let* $\mathsf{SE} = (\mathsf{Enc}_{\mathsf{SE}}, \mathsf{Dec}_{\mathsf{SE}})$ *be an IND-CPA secure symmetric-key encryption. Let* PRF *be a secure pseudorandom function. Then, the DMCFE scheme* $\mathcal{E}'$ *obtained from the above transformation using* $(\mathcal{E}, \mathsf{SE}, \mathsf{PRF})$ *is* weakly function-hiding *against* incomplete adaptive one-challenge *under* static *corruption following Definition 10.*

We state a corollary of Theorem 12 by a direct application of Lemma 6 to achieve *multi-challenge* security for $\mathcal{E}'$ (in the static corruption model), then of Lemma 7 to achieve *fully function-hiding property.*

Corollary 13. *Let* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a DMCFE scheme that is* weakly function-hiding *against* complete adaptive one-challenge *under* static *corruption following Definition 4. Let* $\mathsf{SE} = (\mathsf{Enc}_{\mathsf{SE}}, \mathsf{Dec}_{\mathsf{SE}})$ *be an IND-CPA secure symmetric-key encryption. Let* PRF *be a secure pseudorandom function. Then, there exists a DMCFE scheme constructed from* $(\mathcal{E}, \mathsf{SE}, \mathsf{PRF})$ *that is* function-hiding *against* incomplete adaptive multi-challenge *under* static *corruption following Definition 10.*

Proof (Of Theorem 12 - Sketch). Suppose that there exists a ppt adversary $\mathcal{A}$ breaking the security game $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}w\text{-}any\text{-}fh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ of $\mathcal{E}'$. Then, there exists ppt adversaries $\mathcal{B}, \mathcal{B}', \mathcal{B}''$ that break $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ of $\mathcal{E}$, IND-CPA security of SE, and security of PRF, respectively, and their advantages is lower bounded by that of $\mathcal{A}$, up to a factor polynomial in λ.
The main idea follows the approach of [1, Theorem 4.1], *i.e* we guess if the challenge ciphertext *and* the challenge functional key (in the one-challenge setting) by $\mathcal{A}$ are complete or not. In the former case, we reduce to the security of $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ of $\mathcal{E}$. In the latter case, we apply the IND-CPA security of SE and security of PRF in a hybrid argument.
We define the following games:

Game $\mathsf{G}^*_b(\mathcal{A})$: This game is similar as $\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}w\text{-}any\text{-}fh}\text{-}b}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, for $b \in \{0,1\}$ being the challenge bit, except that our simulator guesses uniformly at random and independently $i^*_e \in \{0,1,\ldots,n\}$ and $i^*_k \in \{0,1,\ldots,n\}$. Intuitively when $i^*_e = 0$ (resp. $i^*_k = 0$), it means the challenge ciphertext (resp. the challenge functional key) by $\mathcal{A}$ are complete; Otherwise, it means the honest i^*_e-th component of the challenge ciphertext (resp. the honest i^*_k-th component of the challenge functional key) is not queried by $\mathcal{A}$. Since we are in the static corruption setting, the simulator can keep track of the queries for honest components since the beginning.
If either i^*_e or i^*_k is not a correct guess, a random result is output. We can verify that

$$\Pr[\mathsf{G}^*_b(\mathcal{A}) = 1] = \frac{1}{(n+1)^2} \Pr[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}w\text{-}any\text{-}fh}\text{-}b}_{\mathcal{E}',\mathcal{F},\mathcal{A}}(1^\lambda) = 1] \ .$$

In the case $i^*_e = i^*_k = 0$, we can rely on the security against *complete* challenges of $\mathcal{E}$. In other words, there exists a ppt adversary $\mathcal{B}$ such that

$$\begin{aligned}
&|\Pr[\mathsf{G}^*_0(\mathcal{A}) = 1 \mid i^*_e = i^*_k = 0] - \Pr[\mathsf{G}^*_1(\mathcal{A}) = 1 \mid i^*_e = i^*_k = 0]| \\
&\quad \leq \left|\Pr[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}\text{-}0}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}\text{-}1}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) = 1]\right| \\
&\quad = \mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \ .
\end{aligned}$$

It remains to deal with the cases of incomplete queries.

Game $\mathsf{H}^{\mathsf{ct}}_{\rho}(\mathcal{A})$ for $\rho \in \{0,\ldots,n\}$: This game is used to argue the difference: for a fixed $j \in [n]$

$$|\Pr[\mathsf{G}^*_0(\mathcal{A}) = 1 \mid i^*_e = j \wedge i^*_k = 0] - \Pr[\mathsf{G}^*_1(\mathcal{A}) = 1 \mid i^*_e = j \wedge i^*_k = 0]| \text{ for } j \in [n] \ .$$

If the guess $i^*_e = j \in [n]$ is correct, then the j-th honest component of the challenge ciphertext is not queried. The game $\mathsf{H}^{\mathsf{ct}}_{\rho}(\mathcal{A})$ is the same as G^*_0, except that all challenge ciphertext components $(i, x^0_i, x^1_i, \mathsf{tag})$ of $i \leq \rho$ is answered by the 0-part of the query, while those of $i > \rho$ is answered by the 1-part of the query. It holds that $\mathsf{H}_0 = \mathsf{G}^*_0$.
For $\rho \in \{0,\ldots,n\}$, we claim that

$$\begin{aligned}&\left|\Pr[\mathsf{H}^{\mathsf{ct}}_{\rho-1}(\mathcal{A}) = 1 \mid i^*_e = j \wedge i^*_k = 0] - \Pr[\mathsf{H}^{\mathsf{ct}}_{\rho}(\mathcal{A}) = 1 \mid i^*_e = j \wedge i^*_k = 0]\right| \\ &\quad \leq \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \ .\end{aligned}$$

This comes from the observation that conditioned on the event $i^*_e = j \wedge i^*_k = 0$ is correct, then the j-th component is not queried to the left-or-right challenge oracle. Moreover, the PRF evaluation $\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{\rho,j}, \mathsf{tag})$ only appears in the query $(\rho, \cdot, \cdot, \mathsf{tag})$ of the challenge ciphertext query. More details follow hereafter.

To go from $\mathsf{H}^{\mathsf{ct}}_{\rho-1}(\mathcal{A})$ to $\mathsf{H}^{\mathsf{ct}}_{\rho}(\mathcal{A})$, firstly, we use the PRF security to switch $\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{\rho,j}, \mathsf{tag})$ to a uniformly random, which is possible because $j \in \mathcal{H}$ and the PRF key is not revealed (conditioned on $i^*_e = j \wedge i^*_k = 0$ is correct). Next, this uniformly random $\oplus$-term makes $K^{\mathsf{ct}}_i(\mathsf{tag})$ uniformly random, and we can apply the IND-CPA security of SE to switch $(\rho, \cdot, \cdot, \mathsf{tag})$ to the encryption of the 1-part of the query. Finally, we come back to $\mathsf{PRF}(\mathsf{k}^{\mathsf{ct}}_{\rho,j}, \mathsf{tag})$ for the query $(\rho, \cdot, \cdot, \mathsf{tag})$ (now encrypting the 1-part), using again the PRF security.

During $(\mathsf{H}^{\mathsf{ct}}_{\rho})^n_{\rho=0}$, we remark that we are using the *weakly* function-hiding condition in $\mathsf{adm}^{\mathsf{any}}$ as we are switching the challenge ciphertext but keep the challenge functional key honestly generated for $(y^0_i)_i$ (as $i^*_k = 0$). After arriving at $\mathsf{H}^{\mathsf{ct}}_n$, where the challenge ciphertext is switched to an encryption of $(x^1_i)_i$, we now need to switch the key to $(y^1_i)_i$ as well, so as to arrive at G^*_1. Because once $\mathcal{A}$ corrupts a client, it receives *both* the encryption key and the secret key of that client, under the condition that $i^*_e = j \wedge i^*_k = 0$ is correct, the secret key sk_j of $j \in \mathcal{H}$ is *not* known to $\mathcal{A}$ either. Therefore, we can switch $\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{\rho',j}, \mathsf{tag})$, for $\rho' \in \{0,\ldots,n\}$, to a uniformly random during the computation of $K^{\mathsf{dk}}_{\rho'}(\mathsf{tag\text{-}f})$, then apply the IND-CPA of SE on the query $(\rho', \cdot, \cdot, \mathsf{tag\text{-}f})$, then switch back to $\mathsf{PRF}(\mathsf{k}^{\mathsf{dk}}_{\rho',j}, \mathsf{tag\text{-}f})$. This incurs another sequence of $n+1$ hybrids on $\rho' \in \{0,\ldots,n\}$ for changing y^0_i into y^1_i.

In the end, for a fixed $j \in [n]$

$$\begin{aligned}&|\Pr[\mathsf{G}^*_0(\mathcal{A}) = 1 \mid i^*_e = j \wedge i^*_k = 0] - \Pr[\mathsf{G}^*_1(\mathcal{A}) = 1 \mid i^*_e = j \wedge i^*_k = 0]| \\ &\quad \leq 2n \cdot \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 4n \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \ .\end{aligned}$$

Game $\mathsf{H}^{\mathsf{dk}}_{\rho}(\mathcal{A})$ for $\rho \in \{0,\ldots,n\}$: This game is used to argue the difference: for a fixed $j \in [n]$

$$|\Pr[\mathsf{G}^*_0(\mathcal{A}) = 1 \mid i^*_e = 0 \wedge i^*_k = j] - \Pr[\mathsf{G}^*_1(\mathcal{A}) = 1 \mid i^*_e = 0 \wedge i^*_k = j]| \ .$$

The argument is totally symmetric to the previous one and we obtain:

$$\begin{aligned}&|\Pr[\mathsf{G}^*_0(\mathcal{A}) = 1 \mid i^*_e = 0 \wedge i^*_k = j] - \Pr[\mathsf{G}^*_1(\mathcal{A}) = 1 \mid i^*_e = 0 \wedge i^*_k = j]| \\ &\quad \leq 2n \cdot \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 4n \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \ .\end{aligned}$$

The case $i_e^* = j_e \wedge i_k^* = j_k$ for $j_e, j_k \in [n]$: By generalizing our above arguments for any fixed $i_k^* \neq 0, j_e \neq 0$:

$$\begin{aligned}
&|\Pr[\mathsf{G}_0^*(\mathcal{A}) = 1 \mid i_e^* = j \wedge i_k^* = j_k] - \Pr[\mathsf{G}_1^*(\mathcal{A}) = 1 \mid i_e^* = j \wedge i_k^* = j_k]| \\
&\quad \leq 2n \cdot \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 4n \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \ .
\end{aligned}$$

We remark that conditioned on $i_e^* = j_e \wedge i_k^* = j_k$ is correct, *i.e.* the honest j_e-th challenge ciphertext component and the honest j_k-th challenge key component are not queried, we can switch on *both* the ct-side (w.r.t (x_i^0, x_i^1)) and the dk-side (w.r.t (y_i^0, y_i^1)) of the ρ-th component, relying on the PRF security as well as the IND-CPA of SE.

Finally, we achieve

$$\begin{aligned}
&\left|\Pr[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}w\text{-}any\text{-}fh\text{-}0}}_{\mathcal{E}',\mathcal{F},\mathcal{A}}(1^\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}w\text{-}any\text{-}fh\text{-}1}}_{\mathcal{E}',\mathcal{F},\mathcal{A}}(1^\lambda) = 1]\right| \\
&= (n+1)^2 \cdot |\Pr[\mathsf{G}_0^*(\mathcal{A}) = 1] - \Pr[\mathsf{G}_1^*(\mathcal{A}) = 1]| \\
&= \left| \sum_{i_e^*=0}^{n} \sum_{i_k^*=0}^{n} (\Pr[\mathsf{G}_0^*(\mathcal{A}) = 1 \mid i_e^* = j_e \wedge i_k^* = j_k] - \Pr[\mathsf{G}_1^*(\mathcal{A}) = 1 \mid i_e^* = j_e \wedge i_k^* = j_k]) \right| \\
&\leq \mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \\
&\quad + \sum_{i_e^*=1}^{n} |\Pr[\mathsf{G}_0^*(\mathcal{A}) = 1 \mid i_e^* = j_e \wedge i_k^* = 0] - \Pr[\mathsf{G}_1^*(\mathcal{A}) = 1 \mid i_e^* = j_e \wedge i_k^* = 0]| \\
&\quad + \sum_{i_k^*=1}^{n} |\Pr[\mathsf{G}_0^*(\mathcal{A}) = 1 \mid i_e^* = 0 \wedge i_k^* = j_k] - \Pr[\mathsf{G}_1^*(\mathcal{A}) = 1 \mid i_e^* = 0 \wedge i_k^* = j_k]| \\
&\quad + \sum_{i_e^*=1}^{n} \sum_{i_k^*=1}^{n} |\Pr[\mathsf{G}_0^*(\mathcal{A}) = 1 \mid i_e^* = j_e \wedge i_k^* = j_k] - \Pr[\mathsf{G}_1^*(\mathcal{A}) = 1 \mid i_e^* = j_e \wedge i_k^* = j_k]| \\
&\leq \mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) + 2n^2 \cdot \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 4n^2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \\
&\quad + 2n^2 \cdot \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 4n^2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \\
&\quad + 2n^3 \cdot \mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 4n^3 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \\
&= \mathbf{Adv}^{\mathsf{1chal\text{-}adap\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) + (4n^2 + 2n^3) \cdot \left(\mathbf{Adv}^{\mathsf{ind\text{-}cpa}}_{\mathsf{SE},\mathcal{B}'}(1^\lambda) + 2 \cdot \mathbf{Adv}^{\mathsf{PRF}}_{\mathcal{B}''}(1^\lambda) \right)
\end{aligned}$$

and the proof is completed. □