Function-Hiding Dynamic Decentralized 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.
Dynamic Decentralized Functional Encryption (DDFE) is the ultimate extension where one can dynamically join the system and the keys and ciphertexts can be built by dynamic subsets of clients.
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.
In this paper, we provide new proof techniques, to analyse our new concrete constructions of function-hiding DMCFE and DDFE for inner products, with strong security guarantees: the adversary can adaptively query multiple challenge ciphertexts and multiple challenge keys. Previous constructions were proven secure in the selective setting only.

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 [SW05, BSW11], 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 [Sha84, Coc01, BF01] or Attribute-Based Encryption [SW05, GPSW06, OSW07, ALdP11, OT12b], and has become a very active domain of research. Abdalla *et al.* [ABDP15] 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 [ALS16, BBL17, CLT18] and quadratic functions [BCFG17, Gay20, AS17, Lin17], or in pushing the studies of new advanced notions [GVW15] as well as the relationship to other notions in cryptography [AJ15, BV15]. 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.* [GGG+14, GKL+13] 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 [DOT18, CDG+18a, CDG+18b, ACF+18, ABKW19, ABG19, LT19, CDSG+20, ACGU20, NPP22].

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.* [CDG+18a]. 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 [CDG+18a], 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 [CDG+18a] 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 [CDG+18a] on both aspects of efficiency as well as security, or by giving generic transformation to (D)MCFE from single-client FE [LT19, ABKW19, ABG19]. All these works follow essentially the syntax of (D)MCFE in [CDG+18a].

Repetitions under One Tag. Involving tags at the time of encryption and key generation restricts that only ciphertexts and partial functional keys having the same tag can be combined in the notion of DMCFE. This raises a natural question: what security can we guarantee when one client uses the same tag on multiple data? We call such multiple usages of the same tag in a DMCFE system *repetitions*. In the formal security model of (D)MCFE in [CDG+18a] and subsequent works [LT19], once the adversary makes a query for (i, tag), further queries for the same pair (i, tag) will be ignored. This means repetitions are not taken into account. The authors of [CDG+18a] argued that it is the responsibility of the users not to use the same tag twice. However, a security notion for

[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.

DMCFE that captures a sense of protection even when repetitions mistakenly/maliciously happen will be preferable, *e.g.* this is indeed studied in some other works [ABKW19, ABG19]. In addition, when repetitions are allowed for ciphertexts, in the security model, MCFE encompasses MIFE, where there is no tag, by just replacing tags by a constant value.

Dynamic Decentralized Functional Encryption. In [CDSG+20], 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. Notably, the setup of DDFE is non-interactive and decentralized, while that of DMCFE can be interactive. When joining a DDFE system, each participant i can run a *local* setup algorithm, which uses some public parameters that is set by a *global* setup algorithm, so as to generate their own secret key sk_i. A set $\mathcal{U}_M$ of clients can use ek_i to independently encrypt their data, contributing to a list of ciphertexts $(\mathsf{ct}_i)_{i\in\mathcal{U}_M}$. Furthermore, a set $\mathcal{U}_K$ of senders can use their sk_i to independently contribute to a list of partial functional keys $(\mathsf{dk}_i)_{i\in\mathcal{U}_K}$. In the end, a DDFE scheme allows aggregating data from different sources by decrypting $(\mathsf{ct}_i)_{i\in\mathcal{U}_M}$ using $(\mathsf{dk}_i)_{i\in\mathcal{U}_K}$, which are fabricated in a completely decentralized manner by clients and senders, while requiring no trusted third party with a master secret key. Being dynamic, a DDFE scheme does not demand in advance a fixed number of clients in $\mathcal{U}_M$ nor a fixed number of senders in $\mathcal{U}_K$.
The authors of [CDSG+20] provided a concrete construction of 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. The notion of DDFE was revisited in a recent work on the notion of *Multi-Party Functional Encryption* (MPFE) [AGT21b].

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 of functional encryption schemes for broader function classes: Lin [Lin17] 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 [Gay20] equally aiming at constructing FE for quadratic functions. With several technical novelties, Agrawal *et al.* [AGT21a, AGT22] were able to generalize the aforementioned constructions to obtain MIFE for quadratic functions.

1.1 Related Works

Other Notions of FE *in the Multi-User Setting.* Many more multi-user FE primitives have been defined, such as *ad hoc multi-input functional encryption* [ACF+20] and *multi-authority attribute-based encryption* [Cha07]. Interestingly, Agrawal *et al.* [AGT21b] 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 [AGT21b], 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 [AGT21b], Agrawal *et al.* proposed a construction for *function-hiding* DDFE from pairings, by specializing the syntax of MPFE, for the function class of inner products. Their scheme achieves indistinguishability-based security in the *random oracle model* (ROM), where multiple challenge ciphertexts and challenge keys must be submitted *selectively* and can handle only *static* corruption of private keys.

Enhancements of FE with Function-Hiding. Bishop *et al.* [BJK15] 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.* [DDM16, DDM17]. This was further improved in terms of efficiency and/or computational hardness assumptions by works of [TAO16, KKS17, KLM+18]. The constructions of [BJK15, DDM16, TAO16] all leverage the power of *dual pairing vector spaces* (DPVSes) developed by Okamoto and Takashima in [OT10, OT12a, OT12b]. In 2017, Lin [Lin17] 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.* [ABDP15] 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.* [ACF+18] were able to construct function-hiding MIFE from standard bilinear maps. As mentioned earlier, Agrawal *et al.* [AGT21b] 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.* [DOT18]. Following the approach of Chotard *et al.* [CDSG+20], they are then able to lift that scheme to function-hiding inner-product DDFE.

(D)MCFE Constructions without Random Oracles. In the seminal work of [CDG+18a], the proposed (D)MCFE schemes were proven secure in the ROM. In [LT19], Libert and Titiu presented the first (D)MCFE that are provably secure under the Learning with Errors assumption in the standard model. Concerning the more general DDFE, all known constructions [CDSG+20, AGT21b] need ROs. Regarding the function-hiding regime, there is a very recent work by Shi and Vanjani [SV23] that presents an FH-MCFE for inner products whose security does not require the ROM. Their techniques include a generic transformation from single-client to multi-client functional encryption, preserving the function-hiding property and leading to the first FH-MCFE with adaptive security without relying on the ROM. We are not aware of any FH-DMCFE or FH-DDFE constructions without using the ROM.

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 [AGT21b]. However, their FH-DDFE yields a construction for FH-DMCFE in an implicit manner. In this paper, we first investigate the problem of constructing *directly* FH-DMCFE for the function class of inner products, taking into account previous (non function-hiding) DMCFE in [CDG+18a, LT19, ABKW19, ABG19] as well as the more general DDFE in [CDSG+20, AGT21b]. Then, using our new FH-DMCFE, we give a candidate for FH-DDFE based on pairings with improved security in comparison with the so far only FH-DDFE from [AGT21b]:

1. 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. Furthermore, our security model for FH-DMCFE allows *repetitions* and thus our construction stays secure even if multiple queries for the same client/sender under the same tag are made by the adversary. And this also encompasses MIFE variants, where some tags are ignored. An overview highlighting our main technical ideas and the details of our achieved security level are given in Section 3.1, while the main constructions are presented in Section 5.
2. More importantly, we leverage our FH-DMCFE to construct an FH-DDFE based on pairings that is provably secure in the ROM against *multiple adaptive* challenge encryption queries and *multiple adaptive* challenge key-generation queries, under *static* corruption. This advances the feasibility of DDFE in terms of functionality and security, as to the best of our knowledge all prior (function-hiding or non function-hiding) DDFE schemes [CDSG+20, AGT21b] achieved only selective security with respect to challenge ciphertexts and challenge keys, with the additional one key-label restriction, which excludes repetitions for key queries. Our key ideas to circumvent the selective security in [CDSG+20, AGT21b] are based on information-theoretic properties of DPVS as well as a different proving approach. A high-level overview and the technical details are given in Sections 3.2 and 6, respectively.
3. Our main technical contribution for achieving *adaptive* security on possibly repetitive challenge ciphertexts and challenge keys is Lemma 12, which is proven and can find other applications in the DPVS setting. This lemma allows us to modify of a family of vectors in a basis, which leads to a *local* change in its products with *some* target family in the dual basis, as long as we do not violate a *global* condition that binds the mentioned vectors to *many* others. All possible repetitions of vectors in a family are considered. For details, see Section 4.
4. As a side contribution, we also propose a new proof technique to analyse a generic conversion from FH-DDFE that is only secure against complete queries to FH-DDFE that can deal with incomplete queries (Remark 7), for both the function-hiding and non function-hiding setting. Tolerating these incomplete queries yields a stronger security model, *e.g.* see [CDG+18b] for more details. Our results then beat the one of [ABG19] in terms of efficiency (linear instead of quadratic ciphertexts) and the one of [CDSG+20] in terms of security (by preserving adaptive security).

2 Preliminaries

For integers m and n with $m < n$, we write $[m; n]$ to denote the set $\{z \in \mathbb{Z} : m \leq z \leq n\}$ and set $[n] \coloneqq [1; n]$. 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 [EHK+13] 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".

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

2.2 Dual Pairing Vector Spaces

Our constructions rely on the *Dual Pairing Vector Spaces* (DPVS) framework in prime-order bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t$ are all written additively. The DPVS technique dates back to the seminal work by Okamoto-Takashima [OT10, OT12a, OT12b] aiming at adaptive security for ABE as well as IBE, together with the *dual system methodology* introduced by Waters [Wat09]. In [LW10], the setting for dual systems is composite-order bilinear groups. Continuing on this line of works, Chen *et al.* [CLL+13] used prime-order bilinear groups under the SXDH assumption. Let us fix $N \in \mathbb{N}$ and consider $\mathbb{G}_1^N$ having N copies of $\mathbb{G}_1$. Any $\mathbf{x} = [\![(m_1, \ldots, m_N)]\!]_1 \in \mathbb{G}_1^N$ is identified as the vector $(m_1, \ldots, m_N) \in \mathbb{Z}_q^N$. 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 $t \cdot \mathbf{x} := [\![t \cdot (m_1, \ldots, m_N)]\!]_1$, where $t \in \mathbb{Z}_q$ and $\mathbf{x} = [\![(m_1, \ldots, m_N)]\!]_1$. The additive inverse of $\mathbf{x} \in \mathbb{G}_1^N$ is defined to be $-\mathbf{x} := [\![(-m_1, \ldots, -m_N)]\!]_1$. 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 $\mathbf{a}_1 := [\![(1, 0 \ldots, 0)]\!]_1, \mathbf{a}_2 := [\![(0, 1, 0 \ldots, 0)]\!]_1, \ldots, \mathbf{a}_N := [\![(0, \ldots, 0, 1)]\!]_1$. 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 $\mathbf{x} = (m_1, \ldots, m_N)_{\mathbf{B}}$ to indicate the representation of $\mathbf{x}$ in the basis $\mathbf{B}$, i.e. $\mathbf{x} = \sum_{i=1}^N m_i \cdot \mathbf{b}_i$. By convention the writing $\mathbf{x} = (m_1, \ldots, m_N)$ 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 $\mathbf{x} = (m_{L[1]}, \ldots, m_{L[\ell]})_{\mathbf{B}[L]}$. Treating $\mathbb{G}_2^N$ similarly, we can furthermore define a product of two vectors $\mathbf{x} = [\![(m_1, \ldots, m_N)]\!]_1 \in \mathbb{G}_1^N, \mathbf{y} = [\![(k_1, \ldots, k_N)]\!]_2 \in \mathbb{G}_2^N$ by $\mathbf{x} \times \mathbf{y} := \prod_{i=1}^N \mathbf{e}(\mathbf{x}[i], \mathbf{y}[i]) = [\![\langle (m_1, \ldots, m_N), (k_1, \ldots, k_N) \rangle]\!]_t$. 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 $B' := (B^{-1})^\top$ 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 $\mathbf{b}_i \times \mathbf{b}_j^* = [\![\delta_{i,j}]\!]_t$ 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 $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ with dual orthogonal bases. We denote by DPVSGen the algorithm that takes as inputs $\mathbb{G}$, a unary 1^N, and some random coins $r \in \{0,1\}^*$, then outputs a pair of random matrices (B, B') that specify dual bases $(\mathbf{B} = [\![B]\!]_1, \mathbf{B}^* = [\![B']\!]_2)$ of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$. Without loss of generality, when the random coins r are fixed we assume that $\mathsf{DPVSGen}(\mathbb{G}, 1^N; r)$ runs in deterministic time $\mathsf{poly}(\log q)$.

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.

2.3 Function-Hiding Dynamic Decentralized Functional Encryption

In this section we recall the notion of *Dynamic Decentralized Functional Encryption* with function privacy, namely *Function-Hiding Dynamic Decentralized Functional Encryption* (FH-DDFE). This notion was defined in [AGT21b, Section 6.1] as a special case of the Multi-Party Functional Encryption notion. In Appendix A.1, we adapt this syntax into the case of FH-DMCFE, which is also a particular case of FH-DDFE, so as to better capture the function-hiding property compared to previous works on non-function hiding DMCFE, *e.g.* [CDG+18a, LT19, ABKW19, ABG19].

Definition 3 (Function-Hiding Dynamic Decentralized Functional Encryption). *Let λ be a security parameter. In the following all domains and sets are indexed by λ, we omit the indices for clarity. Let $\mathcal{K}, \mathcal{M}, \mathsf{ID}$ be a key space, message space, and identity space, where $\mathcal{K} = \mathcal{K}_{\text{pri}} \times \mathcal{K}_{\text{pub}}, \mathcal{M} = \mathcal{M}_{\text{pri}} \times \mathcal{M}_{\text{pub}}$. We consider the function class $\mathcal{F} = \{f : \bigcup_{i\in\mathbb{N}} (\mathsf{ID} \times \mathcal{K})^i \times \bigcup_{i\in\mathbb{N}} (\mathsf{ID} \times \mathcal{M})^i \to \mathcal{R}\}$ for some set $\mathcal{R}$. A* function-hiding dynamic decentralized functional encryption *scheme $\mathcal{E}$ for $\mathcal{F}$ consists of five algorithms* $(\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{GSetup}(1^\lambda)$**:** *Take 1^λ as input and output a set of public parameters* pp*. All other algorithms take* pp *implicitly as inputs.*
$\mathsf{LSetup}(\mathsf{pp})$**:** *Take as input* pp*, output a public key* pk_i *and a secret key* sk_i*.*
$\mathsf{KeyGen}(\mathsf{sk}_i, k = (k_{\text{pri}}, k_{\text{pub}}))$**:** *Given a secret key* sk_i *and $k \in \mathcal{K}$, output* dk_i*.*
$\mathsf{Enc}(\mathsf{sk}_i, m = (m_{\text{pri}}, m_{\text{pub}}))$**:** *Given a secret key* sk_i *and $m \in \mathcal{M}$, output* ct_i*.*
$\mathsf{Dec}((\mathsf{dk}_i)_{i\in\mathcal{U}_K}, (\mathsf{ct}_i)_{i\in\mathcal{U}_M})$**:** *Given $(\mathsf{dk}_i)_{i\in\mathcal{U}_K}, (\mathsf{ct}_i)_{i\in\mathcal{U}_M}$ for $\mathcal{U}_K, \mathcal{U}_M \subseteq \mathsf{ID}$, output either an element in $\mathcal{R}$ or $\perp$.*

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

$$\Pr\left[\, d = f((i,k_i)_{i\in\mathcal{U}_K}, (i,m_i)_{i\in\mathcal{U}_M}) \;\middle|\; \begin{array}{l} (\mathsf{pp}) \leftarrow \mathsf{GSetup}(1^\lambda) \\ (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{LSetup}(\mathsf{pp}) \\ \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{sk}_i, m_i) \\ \mathsf{dk}_i \leftarrow \mathsf{KeyGen}(\mathsf{sk}_i, k_i) \\ d := \mathsf{Dec}((\mathsf{dk}_i)_{i\in\mathcal{U}_K}, (\mathsf{ct}_i)_{i\in\mathcal{U}_M}) \end{array} \right] = 1$$

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

Security. We recall the function-hiding security for DDFE below.

Definition 4 (Function-Hiding Security). *For a* DDFE *scheme $\mathcal{E}$, a function class $\mathcal{F}$ and a* ppt *adversary $\mathcal{A}$ we define the experiment $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ as shown in Figure 1 and set $\mathcal{H} := [n] \setminus \mathcal{C}$. The oracles $\mathcal{O}\mathsf{Enc}$, $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Corrupt}$ can be called in any order and any number of times. We recall that for the queries to $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{KeyGen}$, namely $(i, k_i^{(0)}, k_i^{(1)})$ and $(i, m_i^{(0)}, m_i^{(1)})$, there are private parts $k^{(b)}_{i,\text{pri}}, m^{(b)}_{i,\text{pri}}$ and public parts $k^{(b)}_{i,\text{pub}}, m^{(b)}_{i,\text{pub}}$ in the keys as well as in the messages. We always require $k^{(0)}_{i,\text{pub}} = k^{(1)}_{i,\text{pub}} = k_{\text{pub}}$ and $m^{(0)}_{i,\text{pub}} = m^{(1)}_{i,\text{pub}} = m_{\text{pub}}$ because the public data is not hidden. The adversary $\mathcal{A}$ is* NOT admissible *with respect to $\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{KGen}}$, denoted by $\mathsf{adm}(\mathcal{A}) = 0$, if one of the following holds:*

1. *There exists a tuple $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ such that $i \in \mathcal{C}$ and $m_i^{(0)} \neq m_i^{(1)}$[2], or there exists $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ such that $i \in \mathcal{C}$ and $k_i^{(0)} \neq k_i^{(1)}$.*

[2] This condition was introduced in [CDG+18a] then used in all other works on (D)MCFE [CDG+18a, LT19, ABKW19, ABG19] and later on DDFE [CDSG+20, AGT21b]. We are not aware of any (even non function-hiding) DDFE, or (D)MCFE, scheme in the literature which is proven secure under weaker constraint, *i.e.* more attacks are considered admissible.

2. *(Function-hiding) For* $b \in \{0,1\}$, *there exist* $(i, k_i^{(b)})_{i \in \mathcal{U}_K} \in \bigcup_{i \in \mathbb{N}} (\mathsf{ID} \times \mathcal{K})^i$ *and* $(i, m_i^{(b)})_{i \in \mathcal{U}_M} \in \bigcup_{i \in \mathbb{N}} (\mathsf{ID} \times \mathcal{M})^i$ *such that*
   - $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ *and* $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ *for all* $i \in \mathcal{H}$,
   - $m_i^{(0)} = m_i^{(1)}$ *and* $k_i^{(0)} = k_i^{(1)}$ *for all* $i \in \mathcal{C}$, *and*
   - $f^{(0)}((i, k_i^{(0)})_{i \in \mathcal{U}_K}, (i, m_i^{(0)})_{i \in \mathcal{U}_M}) \neq f^{(1)}((i, k_i^{(1)})_{i \in \mathcal{U}_K}, (i, m_i^{(1)})_{i \in \mathcal{U}_M})$.

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 *if for all* ppt *adversaries* $\mathcal{A}$,

$$\mathbf{Adv}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left| \Pr\left[b' = b\right] - \frac{1}{2} \right|$$

is negligible in λ.

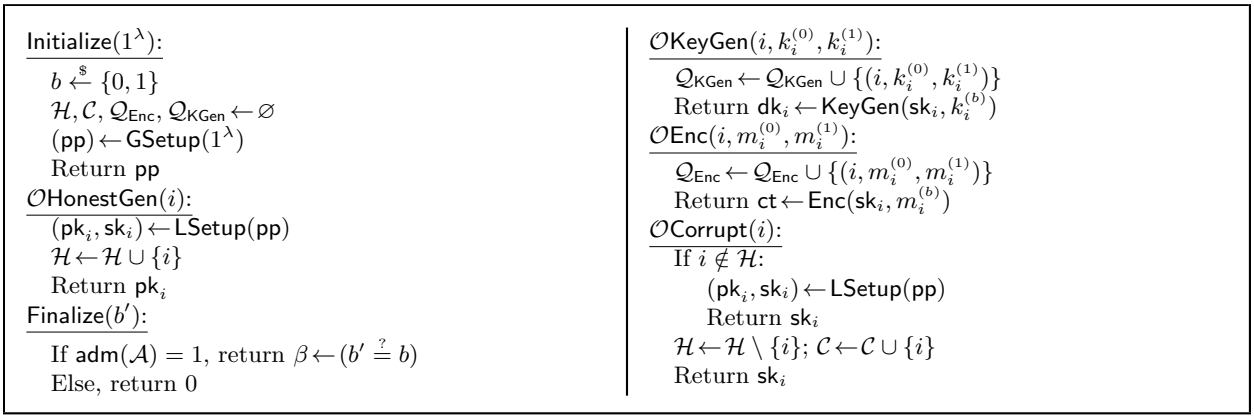

Fig. 1: Security game $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ for Definition 4

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 $\mathbf{Exp}^{\mathsf{stat\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that all queries $\mathcal{C} = \{i\}$ to the oracle $\mathcal{O}$Corrupt must be submitted before Initialize is called. The challenger then performs $\mathsf{LSetup}(\mathsf{pp}) \to (\mathsf{pk}_i, \mathsf{sk}_i)$ for each $i \in \mathcal{C}$ and return $(\mathsf{pk}_i, \mathsf{sk}_i)_{i \in \mathcal{C}}$.
- *Security against Selective Challenges:* The experiment $\mathbf{Exp}^{\mathsf{sel\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that all queries to the oracle $\mathcal{O}$Enc must be submitted before Initialize is called.
- *One-time Security:* The experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that the adversary must declare up front to Initialize additional public information for challenge messages and challenge keys $m_{\mathrm{pub}}, k_{\mathrm{pub}}$ so that:
  - if $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ and $m^{(0)}_{i,\mathrm{pub}} = m^{(1)}_{i,\mathrm{pub}} \neq m_{\mathrm{pub}}$, then $m_i^{(0)} = m_i^{(1)}$,
  - if $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ and $k^{(0)}_{i,\mathrm{pub}} = k^{(1)}_{i,\mathrm{pub}} \neq k_{\mathrm{pub}}$, then $k_i^{(0)} = k_i^{(1)}$.
- *Weakly Function-Hiding:* We can weaken the function-hiding property by changing condition 2 for $\mathsf{adm}(\mathcal{A}) = 0$. More specifically, we replace it by the following condition 2':

  2'. (Weakly Function-Hiding) *For* $b \in \{0,1\}$, *there exist* $(i, k_i^{(b)})_{i \in \mathcal{U}_K} \in \bigcup_{i \in \mathbb{N}} (\mathsf{ID} \times \mathcal{K})^i$ *and* $(i, m_i^{(b)})_{i \in \mathcal{U}_M} \in \bigcup_{i \in \mathbb{N}} (\mathsf{ID} \times \mathcal{M})^i$ *such that*

- $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ *and* $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ *for all* $i \in \mathcal{H}$,
- $m_i^{(0)} = m_i^{(1)}$ *and* $k_i^{(0)} = k_i^{(1)}$ *for all* $i \in \mathcal{C}$, *and*
- $f^{(0)}((i, k_i^{(0)})_{i\in\mathcal{U}_K}, (i, m_i^{(0)})_{i\in\mathcal{U}_M}) \neq f^{(1)}((i, k_i^{(1)})_{i\in\mathcal{U}_K}, (i, m_i^{(1)})_{i\in\mathcal{U}_M})$ OR
  $f^{(0)}((i, k_i^{(0)})_{i\in\mathcal{U}_K}, (i, m_i^{(0)})_{i\in\mathcal{U}_M}) \neq f^{(1)}((i, k_i^{(0)})_{i\in\mathcal{U}_K}, (i, m_i^{(0)})_{i\in\mathcal{U}_M})$ OR
  $f^{(1)}((i, k_i^{(0)})_{i\in\mathcal{U}_K}, (i, m_i^{(0)})_{i\in\mathcal{U}_M}) \neq f^{(1)}((i, k_i^{(1)})_{i\in\mathcal{U}_K}, (i, m_i^{(1)})_{i\in\mathcal{U}_M})$.

The experiment in this weak function-hiding model is denoted by $\mathbf{Exp}^{\mathsf{wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

We define the function family $\mathcal{F}^{\mathsf{IP}}_{N_1,\ldots,N_n}$ of bounded-norm inner-product functionalities that will be used in our FH-DMCFE scheme.

Definition 5 (Inner-Product Functionality). *Let* $\lambda \in \mathbb{N}$. *Let* $B = B(\lambda), n = n(\lambda), N_1(\lambda), \ldots, N_n(\lambda)$ *be polynomials and* $\mathcal{D}_i = [-B; B]^{N_i}$ *for* $i \in [n]$. *We denote by* $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} = \{f_{\mathbf{y}_1,\ldots,\mathbf{y}_n} : \mathbf{y}_1 \in \mathcal{D}_1, \ldots, \mathbf{y}_n \in \mathcal{D}_n\}$ *the family of functions where* $f_{\mathbf{y}_1,\ldots,\mathbf{y}_n} : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathbb{Z}$ *is defined as* $f_{\mathbf{y}_1,\ldots,\mathbf{y}_n}(\mathbf{x}_1, \ldots, \mathbf{x}_n) := \sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i \rangle$.

Below is the dynamic version $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that will be used in our FH-DDFE scheme. Previous works on DDFE [CDSG+20, AGT21b] use the same functionality.

Definition 6 (Dynamic Inner Product Functionality). *Let* $\lambda \in \mathbb{N}$. *We define the* Dynamic Inner-Product Functionality $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ *that contains*

$$f : \bigcup_{i\in\mathbb{N}} \left(\mathsf{ID} \times \left([-B, B]^{N_i} \times \mathcal{K}_{\mathrm{pub}}\right)\right)^i \times \bigcup_{i\in\mathbb{N}} \left(\mathsf{ID} \times \left([-B, B]^{N_i} \times \mathcal{M}_{\mathrm{pub}}\right)\right)^i \to \mathcal{R}$$

where $\mathsf{ID} = \{0,1\}^{\mathsf{poly}(\lambda)}$ *is an identity space,* 2^{ID} *denotes the power set of* ID, $\mathsf{Tag} = \{0,1\}^{\mathsf{poly}(\lambda)}$ *is a tag space,* $\mathcal{K}_{\mathrm{pub}} = \mathcal{M}_{\mathrm{pub}} = 2^{\mathsf{ID}} \times \mathsf{Tag}$, *and* $\mathcal{R} \subset \mathbb{Z}$ *is polynomially large while* $B = B(\lambda), N_i = N_i(\lambda) : \mathbb{N} \to \mathbb{N}$ *are polynomials so that for all* $(i, k_i = (\mathbf{y}_i, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))_{i\in\mathcal{U}_K}$, $(i, m_i = (\mathbf{x}_i, \mathcal{U}_{M,i}, \mathsf{tag}_i))_{i\in\mathcal{U}_M}$

$$f((i, k_i)_{i\in\mathcal{U}_K}, (i, m_i)_{i\in\mathcal{U}_M}) = \begin{cases} \sum_{i\in[n]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle & \text{if condition } (*) \text{ holds} \\ \perp & \text{otherwise} \end{cases}.$$

Condition $(*)$ *holds if:*

- $\mathcal{U}_K = \mathcal{U}_M, |\mathcal{U}_K| = |\mathcal{U}_M| = n$.
- *For all* $i \in \mathcal{U}_K$, $\mathcal{U}_{K,i} = \mathcal{U}_K$; *For all* $i \in \mathcal{U}_M$, $\mathcal{U}_{M,i} = \mathcal{U}_M$.
- *There is* tag-f *such that for all* $i \in \mathcal{U}_K$, $\mathsf{tag\text{-}f}_i = \mathsf{tag\text{-}f}$.
- *There is* tag *such that for all* $i \in \mathcal{U}_M$, $\mathsf{tag}_i = \mathsf{tag}$.

Remark 7. (Security against Complete Challenges) Our FH-DDFE construction in Section 6 for $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ will use an intermediate step proven secure in a weaker model with an additional constraint termed *complete challenges*. Specifically, the experiment $\mathbf{Exp}^{\mathsf{pos\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that we add the following condition *3* for $\mathsf{adm}(\mathcal{A}) = 0$:

3. There exists $m_{\mathrm{pub}} = (\mathcal{U}_M, \mathsf{tag})$ such that a query $\mathcal{O}\mathsf{Enc}(i, (m^{(0)}_{i,\mathrm{pri}}, m_{\mathrm{pub}}), (m^{(1)}_{i,\mathrm{pri}}, m_{\mathrm{pub}}))$ has been asked for some but not all $i \in \mathcal{H} \cap \mathcal{U}_M$, or there exists $k_{\mathrm{pub}} = (\mathcal{U}_K, \mathsf{tag\text{-}f})$ such that a query $\mathcal{O}\mathsf{KeyGen}(i, (k^{(0)}_{i,\mathrm{pri}}, k_{\mathrm{pub}}), (k^{(1)}_{i,\mathrm{pri}}, k_{\mathrm{pub}}))$ has been asked for some but not all $i \in \mathcal{H} \cap \mathcal{U}_K$.

Lemma 8 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 8. *Let* $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a DDFE scheme for the function class* $\mathcal{F}$*. If* $\mathcal{E}$ *is single-challenge weakly function-hiding, then it is also weakly function-hiding. More specifically, for any* ppt *adversary* $\mathcal{A}$*, there exists a* ppt *algorithm* $\mathcal{B}$ *such that*

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

where q_e *and* q_k *denote the maximum numbers of different* m_{pub} *and* k_{pub} *that* $\mathcal{A}$ *can query to* $\mathcal{O}\mathsf{Enc}$ *and* $\mathcal{O}\mathsf{KeyGen}$ *respectively, and* $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}, \mathsf{pos}\}$.

The works of [LV16] and [ACF+18] 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 9, is also applicable in the case of FH-DDFE. The proof is given in Appendix B.2.

Lemma 9. *If there exists a weakly function-hiding DDFE scheme* $\mathcal{E}$ *for* $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$*, then there exists a (fully) function-hiding DDFE scheme* $\mathcal{E}'$ *for* $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$*. More precisely, for any* ppt *adversary* $\mathcal{A}$*, there exists a* ppt *algorithm* $\mathcal{B}$ *such that*

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

where $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}, \mathsf{1chal}, \mathsf{pos}\}$.

All-or-Nothing Encapsulation (AoNE). The notion of AoNE is a particular functionality of DDFE introduced by Chotard *et al.* [CDSG+20]. In the transformation of [CDG+18b, Section 5.2], AoNE appears under the name *Secret Sharing Layer (SSL)*. In [AGT21b], AoNE also serves as a building block for their FH-DDFE scheme. In this paper we follow the syntax of [AGT21b] and regard AoNE as an FH-DDFE for a specific functionality. We remark that in [CDSG+20], a separate indistinguishability-based security notion is defined, but because there is no concepts of keys in AoNE, the security notion of [CDSG+20] is equivalent to the function-hiding security from Definition 4 when viewing AoNE as an FH-DDFE.

Definition 10 (All-or-Nothing Encapsulation). *Let* $\lambda \in \mathbb{N}$*. We define the* All-or-Nothing Encapsulation *(AoNE) as FH-DDFE schemes for the functionality that contains*

$$f : \varnothing \times \bigcup_{i \in \mathbb{N}} (\mathsf{ID} \times (\mathcal{D} \times \mathcal{M}_{\mathrm{pub}}))^i \to \mathcal{R}$$

where $\mathsf{ID} = \{0,1\}^{\mathsf{poly}(\lambda)}$ *is an identity space,* 2^{ID} *denotes the power set of* ID*,* $\mathsf{Tag} = \{0,1\}^{\mathsf{poly}(\lambda)}$ *is a tag space,* $\mathcal{M}_{\mathrm{pub}} = 2^{\mathsf{ID}} \times \mathsf{Tag}$*, and* $\mathcal{R} = \{0,1\}^*$ *so that for all* $(i)_{i \in \mathcal{U}_K}$*,* $(i, m_i = (x_i, \mathcal{U}_{M,i}, \mathsf{tag}_i))_{i \in \mathcal{U}_M}$

$$f((i)_{i \in \mathcal{U}_K}, (i, m_i)_{i \in \mathcal{U}_M}) = \begin{cases} (x_i)_{i \in \mathcal{U}_M} & \text{if condition } (*) \text{ holds} \\ \bot & \text{otherwise} \end{cases} .$$

Condition $(*)$ *holds if:*

- *For all* $i \in \mathcal{U}_M$*,* $\mathcal{U}_{M,i} = \mathcal{U}_M$*.*
- *There is* tag *such that for all* $i \in \mathcal{U}_M$*,* $\mathsf{tag}_i = \mathsf{tag}$*.*

From [CDSG+20], it holds that AoNE can be constructed generically from identity-based encryption, or from bilinear maps under the *Decisional Bilinear Diffie-Hellman (DBDH)* assumption. The first construction is secure in the standard model, but ciphertexts have size linear in $|\mathcal{U}_M|$. On the other hand, the size of ciphertexts in the construction from DBDH is are independent of $|\mathcal{U}_M|$, but the security proof resorts to the ROM.

3 Technical Overview

3.1 Function-Hiding DMCFE for Inner Products

Specializing FH-DMCFE using FH-DDFE. In [CDG+18a] and its follow-up works [CDG+18b, ABKW19, ABG19, LT19, CDSG+20] on DMCFE, the syntax of DMCFE is given by (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. 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. 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 needs certain refinements to be able to capture the privacy of functions. Our very first step is to specialize the FH-DDFE notion from Definition 3 to the case of FH-DMCFE. The complete definition as well as detailed security model for FH-DMCFE, because the algorithms' names are modified from the FH-DDFE, are given in Appendix A.1.
As an intermediate step, we focus on constructing an FH-DMCFE scheme whose proof of security is done in a model with certain restrictions. Specifically, we consider the case where the adversary is allowed to ask only *complete* challenges, *one adaptive challenge* query, under *static* corruption, and all their queries, with possibly *repetitions*, are subject to the *weakly* function-hiding condition:

- *Weakly function-hiding (Condition 5 in Definition 18):* All queries satisfy $F^{(0)}(x_1^{(0)}, \ldots, x_n^{(0)}) = F^{(1)}(x_1^{(0)}, \ldots, x_n^{(0)}) = F^{(1)}(x_1^{(1)}, \ldots, x_n^{(1)})$.
- *Complete queries constraint (Condition 4 in Definition 18):* If there exists a ciphertext (or key) query for any honest component w.r.t some tag tag (or tag-f), then all honest ciphertext (or key) components must be queried at least once for this tag.
- *One challenge constraint (Condition 3 in Definition 18):* There exists only one tag^* to the encryption oracle $\mathcal{O}\mathsf{Enc}$ having $x_i^{(0)} \neq x_i^{(1)}$, while for other $\mathsf{tag} \neq \mathsf{tag}^*$ it holds that $x_i^{(0)} = x_i^{(1)}$. Similarly, there exists only one $\mathsf{tag\text{-}f}^*$ to the key generation oracle $\mathcal{O}\mathsf{KeyGen}$ having $y_i^{(0)} \neq y_i^{(1)}$, while for other $\mathsf{tag\text{-}f} \neq \mathsf{tag\text{-}f}^*$ it holds that $y_i^{(0)} = y_i^{(1)}$.
- *Static corruption (Condition 1 in Definition 18):* All corruption queries must be declared up front.

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 5. We use DPVSes in the bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$. Our function class of interest is for computing inner products $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ that is defined in Definition 5. Each sender can use their secret key sk_i to independently generate a partial functional decryption key for $\mathbf{y}_i \in \mathbb{Z}_q^{N_i}$, and each client can use their encryption key ek_i to independently encrypt their data $\mathbf{x}_i \in \mathbb{Z}_q^{N_i}$. 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 generation tags. Given tag for encryption and tag-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 tag-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 $[\![\mu]\!]_2$ to randomize a pre-generated secret sharing $(\tilde{s}_i)_i$. Additionally, we need more coordinates for the goal of *repeated 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:

$$\begin{array}{lllllllll} \mathbf{c}_i = (& \mathbf{x}_i & \omega & 0^{N_i} & 0^{N_i} & t_i & 0 & \rho_i & \text{aux. 0-coords} \quad)_{\mathbf{B}_i} \\ \mathbf{d}_i = (& \mathbf{y}_i & s_i & 0^{N_i} & 0^{N_i} & \mu & \pi_i & 0 & \text{aux. 0-coords} \quad)_{\mathbf{B}_i^*} \end{array}$$

where $\pi_i, \rho_i \xleftarrow{\$} \mathbb{Z}_q$ are randomness. The auxiliary coordinates that are not used in real ciphertexts and keys, thus containing 0's, are for proving security.
We recall that our one-challenge security notion means there exists only one tag^* to the encryption oracle $\mathcal{O}\mathsf{Enc}$ having $(\mathbf{x}_i^{(0,j)})_i \neq (\mathbf{x}_i^{(1,j)})_i$, while for other $\mathsf{tag}_\ell \neq \mathsf{tag}^*$ it holds that $(\mathbf{x}_{\ell,i}^{(0,j)})_{\ell,i} = (\mathbf{x}_{\ell,i}^{(1,j)})_i$. We denote by $(\mathbf{c}_{\ell,i}^{(j')})_i$ the ciphertext components for $(\mathbf{x}_{\ell,i}^{(j')})_i$ under these non-challenge tag_ℓ, with possible different repetitions $\mathbf{x}_{\ell,i}^{(j')}$ for different j'. In the same manner, there exists only one $\mathsf{tag\text{-}f}^*$ to the key-generation oracle $\mathcal{O}\mathsf{KeyGen}$ having $(\mathbf{y}_i^{(0,\tilde{j})})_i \neq (\mathbf{y}_i^{(1,\tilde{j})})_i$, while for other $\mathsf{tag\text{-}f}_k \neq \mathsf{tag\text{-}f}^*$ it holds that $(\mathbf{y}_{k,i}^{(0,\tilde{j})})_{k,i} = (\mathbf{y}_{k,i}^{(1,\tilde{j})})_i$. We denote by $(\mathbf{d}_{k,i}^{(\tilde{j}')})_i$ the key components for $(\mathbf{y}_{k,i}^{(\tilde{j}')})_i$ under these non-challenge $\mathsf{tag\text{-}f}_k$, with possible different repetitions $\mathbf{y}_{k,i}^{(\tilde{j}')}$ for different $\tilde{j}'$. To summarize, starting from the game with the challenge bit $b \xleftarrow{\$} \{0,1\}$, an adversary will obtain:

$$\begin{array}{lllllllll} \mathbf{c}_{\ell,i}^{(j')} = (& \mathbf{x}_{\ell,i}^{(j')} & \omega & 0^{N_i} & 0^{N_i} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & \text{aux. 0-coords} \quad)_{\mathbf{B}_i} \\ \mathbf{c}_i^{(j)} = (& \mathbf{x}_i^{(b,j)} & \omega & 0^{N_i} & 0^{N_i} & t_i & 0 & \rho_i^{(j)} & \text{aux. 0-coords} \quad)_{\mathbf{B}_i} \\ \mathbf{d}_i^{(\tilde{j})} = (& \mathbf{y}_i^{(b,\tilde{j})} & s_i & 0^{N_i} & 0^{N_i} & \mu & \pi_i^{(\tilde{j})} & 0 & \text{aux. 0-coords} \quad)_{\mathbf{B}_i^*} \\ \mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i}^{(\tilde{j}')} & s_{k,i} & 0^{N_i} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \text{aux. 0-coords} \quad)_{\mathbf{B}_i^*} \end{array} .$$

The challenges $(\mathbf{c}_i^{(j)}, \mathbf{d}_i^{(\tilde{j})})_i$ are queried adaptively, while the set of corrupted i is declared up front. We allow repetitions at a position i under the same tag, indexed by (j, j') for ciphertext tags $(\mathsf{tag}^*, \mathsf{tag}_\ell)$ and by $(\tilde{j}, \tilde{j}')$ for functional key tags $(\mathsf{tag\text{-}f}^*, \mathsf{tag\text{-}f}_k)$, in that order. Our goal is to switch $(\mathbf{x}_i^{(b,j)}, \mathbf{y}_i^{(b,\tilde{j})})_i$ in $(\mathbf{c}_i, \mathbf{d}_i)_i$ to $(\mathbf{x}_i^{(1,j)}, \mathbf{y}_i^{(1,\tilde{j})})_i$ so that the game does not depend on $b \xleftarrow{\$} \{0,1\}$ and the adversary's advantage becomes 0. We recall that the secret sharings $(s_i, t_i, s_{k,i}, t_{\ell,i})_{i,k,\ell}$ depend only on the tags and a fixed secret sharing generated from setup time, hence they are the same across repetitions under the same tag.

Concrete Interpretation of the Function-Hiding Admissibility. A step of vital importance in our proof is a concrete interpretation of the *admissible* adversaries against an $\mathsf{FH\text{-}DMCFE}$ for $\mathcal{F}_{N_1,\ldots,N_n}^{\mathsf{ip}}$. The general definition of admissible adversaries for $\mathsf{FH\text{-}DDFE}$ is given in Definition 4, and we specialized these conditions in the case of $\mathsf{FH\text{-}DMCFE}$ in Definition 18:

- Condition 1 of Definition 18 restricts that if a client $i \in [n]$ is corrupted by the adversary then the challenge queries at i must be on equal messages $\mathbf{x}_i^{(0,j)} = \mathbf{x}_i^{(1,j)}$ and function parameters $\mathbf{y}_i^{(0,\tilde{j})} = \mathbf{y}_i^{(1,\tilde{j})}$. This condition is inspired from a similar admissibility condition for $(\mathsf{D})\mathsf{MCFE}$ from the works [CDG+18a, CDSG+20] and the function-hiding admissibility condition from [AGT22].
- Finally, condition 2 dictates that: let $b \xleftarrow{\$} \{0,1\}$ be the challenge bit, then

$$f_{\mathbf{y}_1^{(0,\tilde{j}_{i_1})},\ldots,\mathbf{y}_n^{(0,\tilde{j}_{i_1})}}(\mathbf{x}_1^{(0,j_1)},\ldots,\mathbf{x}_n^{(0,j_n)}) = f_{\mathbf{y}_1^{(1,\tilde{j}_{i_1})},\ldots,\mathbf{y}_n^{(1,\tilde{j}_{i_1})}}(\mathbf{x}_1^{(1,j_1)},\ldots,\mathbf{x}_n^{(1,j_n)}) \tag{1}$$

  for *all* possible combinations $(\mathbf{x}_1^{(b,j_1)},\ldots,\mathbf{x}_n^{(b,j_n)})$ as well as *all* possible combinations $(\mathbf{y}_1^{(b,\tilde{j}_1)},\ldots,\mathbf{y}_n^{(b,\tilde{j}_n)})$, over the potentially repetitive ciphertext challenges $(i, \mathsf{tag}^*, \mathbf{x}_i^{(0,j)}, \mathbf{x}_i^{(1,j)})$ and $(i, \mathsf{tag\text{-}f}^*, \mathbf{y}_i^{(0,\tilde{j})}, \mathbf{y}_i^{(1,\tilde{j})})$ at each position i. We emphasize that for each i, the number of ciphertext repetitions J_i ($\tilde{J}_i$ for keys) might differ and the combination is specified by $(j_1,\ldots,j_n) \in [J_1] \times \cdots \times [J_n]$, or $(\tilde{j}_1,\ldots,\tilde{j}_n) \in [\tilde{J}_1] \times \cdots \times [\tilde{J}_n]$ for keys. Moreover, condition 2 implies that for each client $i \in [n]$, for all queries $(i, \mathsf{tag}^*, \mathbf{x}_i^{(0,j)}, \mathbf{x}_i^{(1,j)})$ and $(i, \mathsf{tag\text{-}f}^*, \mathbf{y}_i^{(0,\tilde{j})}, \mathbf{y}_i^{(1,\tilde{j})})$, it must hold that:

$$\langle \mathbf{x}_i^{(0,j)}, \mathbf{y}_i^{(0,\tilde{j})} \rangle - \langle \mathbf{x}_i^{(1,j)}, \mathbf{y}_i^{(1,\tilde{j})} \rangle = \langle \mathbf{x}_i^{(0,j')}, \mathbf{y}_i^{(0,\tilde{j}')} \rangle - \langle \mathbf{x}_i^{(1,j')}, \mathbf{y}_i^{(1,\tilde{j}')} \rangle \tag{2}$$

for every repetitions $j, j' \in [J_i]$ and every $\tilde{j}, \tilde{j}' \in [\tilde{J}_i]$.

This concrete interpretation will play a crucial role in the proof of Theorem 13 for function-hiding security of our FH-DMCFE scheme.

A Key Lemma - Secret Sharing Swapping. Back to the *weakly* function-hiding of our FH-DMCFE scheme, the condition (1) from fully function-hiding becomes

$$\begin{aligned} f_{\mathbf{y}_1^{(0,\tilde{j}_i)},\ldots,\mathbf{y}_n^{(0,\tilde{j}_i)}}(\mathbf{x}_1^{(0,j_1)},\ldots,\mathbf{x}_n^{(0,j_n)}) &= f_{\mathbf{y}_1^{(1,\tilde{j}_i)},\ldots,\mathbf{y}_n^{(1,\tilde{j}_i)}}(\mathbf{x}_1^{(0,j_1)},\ldots,\mathbf{x}_n^{(0,j_n)}) \\ &= f_{\mathbf{y}_1^{(1,\tilde{j}_i)},\ldots,\mathbf{y}_n^{(1,\tilde{j}_i)}}(\mathbf{x}_1^{(1,j_1)},\ldots,\mathbf{x}_n^{(1,j_n)}) \end{aligned} \tag{3}$$

over all combinations of repetitive components among ciphertexts and keys. The same adaptation also applies to condition (2) in this weakly function-hiding setting:

$$\begin{cases} \langle \mathbf{x}_i^{(0,j)}, \mathbf{y}_i^{(0,\tilde{j})} \rangle - \langle \mathbf{x}_i^{(0,j)}, \mathbf{y}_i^{(1,\tilde{j})} \rangle &= \langle \mathbf{x}_i^{(0,j')}, \mathbf{y}_i^{(0,\tilde{j}')} \rangle - \langle \mathbf{x}_i^{(0,j')}, \mathbf{y}_i^{(1,\tilde{j}')} \rangle \\ \langle \mathbf{x}_i^{(0,j)}, \mathbf{y}_i^{(0,\tilde{j})} \rangle - \langle \mathbf{x}_i^{(1,j)}, \mathbf{y}_i^{(1,\tilde{j})} \rangle &= \langle \mathbf{x}_i^{(0,j')}, \mathbf{y}_i^{(0,\tilde{j}')} \rangle - \langle \mathbf{x}_i^{(1,j')}, \mathbf{y}_i^{(1,\tilde{j}')} \rangle \end{cases} \tag{4}$$

for each client $i \in [n]$ and for every repetitions $j, j' \in [J_i]$ and every $\tilde{j}, \tilde{j}' \in [\tilde{J}_i]$. Condition (3) implies that, for the challenge bit $b \xleftarrow{\$} \{0,1\}$

$$\begin{aligned} f_{\mathbf{y}_1^{(b,\tilde{j}_1)}-\mathbf{y}_1^{(1,\tilde{j}_1)},\ldots,\mathbf{y}_n^{(b,\tilde{j}_n)}-\mathbf{y}_n^{(1,\tilde{j}_n)}}(\mathbf{x}_1^{(b,j_1)},\ldots,\mathbf{x}_n^{(b,j_n)}) &\overset{(*)}{=} \sum_{i=1}^{H} \langle \mathbf{x}_i^{(b,j_i)}, \mathbf{y}_i^{(b,\tilde{j}_i)} - \mathbf{y}_i^{(1,\tilde{j}_i)} \rangle \\ &= 0 \end{aligned}$$

where H denotes the number of honest i and $(*)$ comes from condition 1 of Definition 18 on corrupted i. This leads to our idea to treat each term $\langle \mathbf{x}_i^{(b,j_i)}, \mathbf{y}_i^{(b,\tilde{j}_i)} - \mathbf{y}_i^{(1,\tilde{j}_i)} \rangle$ as *some i-th share in a secret sharing of* 0, which depends on some adversarial vectors $(\mathbf{x}_i^{(b,j_i)}, \mathbf{y}_i^{(b,\tilde{j}_i)})_{i=1}^n$. By combining with condition (4), we prove a computational lemma in the DPVS setting that states:

> Given the adversarially chosen vectors $(\mathbf{x}_i^{(b,j_i)}, \mathbf{y}_i^{(b,\tilde{j}_i)})_{i=1}^n$, under appropriately crafted random dual bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ for DPVS and conditions (3) as well as (4), given $\mathbf{c}_i$-vectors in $\mathbf{B}_i$ encoding $\mathbf{x}_i^{(b,j_i)}$ and $\mathbf{d}_i$-vectors encoding $(\mathbf{y}_i^{(b,\tilde{j}_i)}, \mathbf{y}_i^{(1,\tilde{j}_i)})$, under the SXDH assumption, we are able to modify $\mathbf{c}_i$-vectors and $\mathbf{d}_i$-vectors so that $\mathbf{c}_i \times \mathbf{d}_i$ changes from $\langle \mathbf{x}_i^{(b,j_i)}, \boxed{\mathbf{y}_i^{(b,\tilde{j}_i)}} \rangle$ to $\langle \mathbf{x}_i^{(b,j_i)}, \boxed{\mathbf{y}_i^{(1,\tilde{j}_i)}} \rangle$.

The formal statement is presented in Lemma 12. We remark that because we want to apply this lemma in the context with multiple families of vectors $(\mathbf{c}_i^{(j)}, \mathbf{d}_i^{(\tilde{j})})_i$, where at a position $i \in [n]$ there might be repetitions of $\mathbf{c}_i^{(j)}$ or $\mathbf{d}_i^{(\tilde{j})}$, Lemma 12 also takes into account those details. An overview of the proof for Lemma 12 is given in Section 4, all relevant details can be found in Appendix B.3.

Finishing the Proof. Being armed with Lemma 12, we are ready to tackle the security proof of our FH-DMCFE from Section 5. In the following the changes in vectors are put in boxed. First of all, because we consider only *static* corruption, the changes are made only for honest position i, whose $(\mathsf{ek}_i, \mathsf{sk}_i)$ are never revealed to the adversary.
After some pre-processing steps to switch the secret sharings $(s_i, t_i, s_{k,i}, t_{\ell,i})_{i,k,\ell}$ from being shifted by the hash values of tags to being independently randomly chosen secret sharing of 0, we begin by exploiting the computational *subspace indistinguishability* of DPVS to prepare the vectors:

$$\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& \mathbf{x}_{\ell,i}^{(j')} & \omega & 0^{N_i} & 0^{N_i} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{c}_{i}^{(j)} = (& \mathbf{x}_{i}^{(b,j)} & \omega & 0^{N_i} & 0^{N_i} & t_i & 0 & \rho_{i}^{(j)} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{d}_{i}^{(\tilde{j})} = (& \mathbf{y}_{i}^{(b,\tilde{j})} & s_i & \boxed{\mathbf{y}_{i}^{(1,\tilde{j})}} & 0^{N_i} & \mu & \pi_{i}^{(\tilde{j})} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i}^{(\tilde{j}')} & s_{k,i} & \boxed{\mathbf{y}_{k,i}^{(\tilde{j}')}} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*}
\end{array}.$$

This computational property of DPVS was introduced in the seminal works by Okamoto and Takashima [OT10, OT12a, OT12b]. Under the DDH assumption, this subspace indistinguishability allows us to introduce chosen values into certain coordinates of *some* vectors in a basis (in this case $\mathbf{d}_i^{(\tilde{j})}$ and $\mathbf{d}_{k,i}^{(\tilde{j}')}$ in $\mathbf{B}_i^*$), as long as the corresponding coordinates in *all* given vectors in the *dual* basis are 0 (in this case $\mathbf{c}_i^{(j)}$ and $\mathbf{c}_{\ell,i}^{(j')}$ in $\mathbf{B}_i$). Next, we apply Lemma 12:

$$\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& \boxed{0^{N_i}} & \omega & \boxed{\mathbf{x}_{\ell,i}^{(j')}} & 0^{N_i} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{c}_{i}^{(j)} = (& \boxed{0^{N_i}} & \omega & \boxed{\mathbf{x}_{i}^{(b,j)}} & 0^{N_i} & t_i & 0 & \rho_{i}^{(j)} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{d}_{i}^{(\tilde{j})} = (& \mathbf{y}_{i}^{(b,\tilde{j})} & s_i & \mathbf{y}_{i}^{(1,\tilde{j})} & 0^{N_i} & \mu & \pi_{i}^{(\tilde{j})} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i}^{(\tilde{j}')} & s_{k,i} & \mathbf{y}_{k,i}^{(\tilde{j}')} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*}
\end{array}.$$

The applications are performed via a sequence of hybrids over distinct tags, including the challenge tag^* and non-challenge tag_ℓ, that are queried to $\mathcal{O}\mathsf{Enc}$, up to repetitions j, j' at each position i. As an example, we can verify that the interpreted admissibility as conditions (3) and (4), together with the static corruption ensuring all basis vectors of an honest i are hidden, satisfy the hypothesis of Lemma 12 and permit us to swap $\mathbf{x}_i^{(b,j)}$ for each honest i. The swapping $\mathbf{x}_{\ell,i}^{(j')}$ can be checked similarly. We then employ again the subspace indistinguishability

$$\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& 0^{N_i} & \omega & \mathbf{x}_{\ell,i}^{(j')} & \boxed{\mathbf{x}_{\ell,i}^{(j')}} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{c}_{i}^{(j)} = (& 0^{N_i} & \omega & \mathbf{x}_{i}^{(b,j)} & \boxed{\mathbf{x}_{i}^{(1,j)}} & t_i & 0 & \rho_{i}^{(j)} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{d}_{i}^{(\tilde{j})} = (& \mathbf{y}_{i}^{(b,\tilde{j})} & s_i & \mathbf{y}_{i}^{(1,\tilde{j})} & 0^{N_i} & \mu & \pi_{i}^{(\tilde{j})} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i}^{(\tilde{j}')} & s_{k,i} & \mathbf{y}_{k,i}^{(\tilde{j}')} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*}
\end{array}$$

and perform a sequence of hybrids over distinct tags to $\mathcal{O}\mathsf{KeyGen}$ to apply Lemma 12 for swapping the keys' contents:

$$\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& \boxed{0^{N_i}} & \omega & \mathbf{x}_{\ell,i}^{(j')} & \mathbf{x}_{\ell,i}^{(j')} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{c}_{i}^{(j)} = (& \boxed{0^{N_i}} & \omega & \mathbf{x}_{i}^{(b,j)} & \mathbf{x}_{i}^{(b,j)} & t_i & 0 & \rho_{i}^{(j)} & \text{aux. 0-coords} &)_{\mathbf{B}_i} \\
\mathbf{d}_{i}^{(\tilde{j})} = (& \mathbf{y}_{i}^{(b,\tilde{j})} & s_i & \boxed{0^{N_i}} & \boxed{\mathbf{y}_{i}^{(1,\tilde{j})}} & \mu & \pi_{i}^{(\tilde{j})} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i}^{(\tilde{j}')} & s_{k,i} & \boxed{0^{N_i}} & \boxed{\mathbf{y}_{k,i}^{(\tilde{j}')}} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \text{aux. 0-coords} &)_{\mathbf{B}_i^*}
\end{array}.$$

The very last step uses subspace indistinguishability for cleaning the first and third columns of ciphertexts as well as keys, leading to a game independent of the challenge $b \xleftarrow{\$} \{0,1\}$.

Removing the usage of random oracles in FH-DMCFE/FH-DDFE. In a concurrent work, Shi and Vanjani [SV23] propose the first FH-MCFE for inner products using pairings that is provably secure in the standard model. An important building-block in their FH-MCFE is the primitive of *correlated pseudorandom functions* (Cor-PRFs) from [BIK+17, ABG19, SW21]. In the setting of MCFE, the functional key generation is centralized and the secret sharing of 0 (similar to our $(s_i)_i$) can be generated using a maser secret key, while the counterpart in ciphertexts (similar to our $(t_i)_i$) can be computed independently by each client using a Cor-PRF, whose key is given in ek_i. This helps remove the reliability on the RO to randomize a pre-determined secret sharing and allows proving security in the standard model. However, all these secret shares, $(s_i)_i$ or $(t_i)_i$, must be multiplied by a fixed randomness at decryption time (similar to our ω or μ respectively) so as to enable correct decryption. Moving to the decentralized setting of DMCFE, or the dynamic setting of DDFE, we

believe that it will be delicate to make ciphertext and key components agree on such a randomness while using Cor-PRF.

3.2 Function-Hiding DDFE for Inner Products

In this section, we describe the main ideas in the construction of our FH-DDFE scheme for the inner product functionality. Our goal is to achieve *adaptive* security for both encryption and key-generation queries under *static* corruption. As a starting point, we attempt to follow the blueprint of [AGT21b]. They provide a construction of a function-hiding DDFE scheme proven secure under selective key-generation and encryption queries as well as static corruption. Furthermore, their proof uses an additional assumption about the adversary termed *one key-label restriction*.[3] The construction proceeds in two steps. First, the authors build an FH-MCFE scheme secure against complete challenges. Subsequently, they lift this scheme to FH-DDFE. Inspecting their transformation, we identify the following key challenges:

1. *Decentralized generation of decryption keys:* In a DMCFE or DDFE, several users generate partial decryption keys and decryption is only possible when all parts are available. The difficulty lies in the fact that the adversary can corrupt users and learn their secret keys. However, this should not allow generating decryption keys for honest users.
2. *Non-interactive generation of secret keys:* Being dynamic, DDFE aims at allowing users to join the system at any time without interaction with the other users, whereas the number of users is fixed in (D)MCFE.
3. *Security against incomplete queries:* The security proof for the MCFE of [AGT21b] assumes that for each tag tag, the adversary either does not ask encryption queries for any honest client, or it asks at least one query for each honest client. In this case, we say that the adversary asks only *complete queries*. The security of the final DDFE scheme does not rely on this assumption anymore.

In the remainder of this section, we discuss how the authors of [AGT21b] address these challenges and explain our modifications that allow us to prove adaptive security without the one key-label restriction.

Challenge 1: Decentralized Generation of Decryption Keys. The authors of [AGT21b] deal with the first challenge in a natural way by employing a concrete MCFE scheme whose decryption keys consist of n shares, each corresponding to one client. In our conversion, we formalize this fact by directly starting from a DMCFE scheme. As the DDFE scheme essentially inherits the security level of the underlying (D)MCFE, it is important to compare the security levels of these schemes.
Most importantly, the function-hiding MCFE scheme in [AGT21b] is only selectively secure, whereas our DMCFE enjoys adaptive security. Although the construction was already described in Section 3.1, it is instructive to compare it explicitly with the MCFE of [AGT21b]. The starting points are very similar. Given a tuple $(i, \mathbf{x}_i, \mathsf{tag})$, the encryption algorithms of both schemes define an extended vector of the form $\hat{\mathbf{x}}_i = (\mathbf{x}_i, \mathsf{tag}, \cdots)$ and return an encryption of $\hat{\mathbf{x}}_i$ using a single-input function-hiding FE scheme for inner products. Similarly, partial keys for a tuple $(i, \mathbf{y}_i, \mathsf{tag\text{-}f})$ are created by choosing a random secret sharing $(s_i)_{i\in[n]}$ of 0, defining $\hat{\mathbf{y}}_i = (\mathbf{y}_i, s_i, \cdots)$ and returning a key for $\hat{\mathbf{y}}_i$ that is generated using the same single-input FE scheme.[4]

[3] The one key-label restriction is an additional constraint on the adversary in the security game of FH-DDFE for the inner-product functionality $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$. Specifically, an adversary satisfies the one key-label restriction if their queries satisfy the following additional condition: The query $\mathcal{O}\mathsf{KeyGen}(i, (k^0_{\mathrm{pri}}, k_{\mathrm{pub}}), (k^1_{\mathrm{pri}}, k_{\mathrm{pub}}))$ for $k_{\mathrm{pub}} = (*, \mathsf{tag\text{-}f})$ is made only once for each pair $(i, \mathsf{tag\text{-}f})$.

[4] We recall that keys in the MCFE of [AGT21b] implicitly have this modular structure, even if the notion of partial key is not properly defined for MCFE schemes in general.

The admissibility for adversaries in the function-hiding security game for (D)MCFE specifies global conditions of the form $\sum_{i=1}^{n}\langle \mathbf{x}_i^0, \mathbf{y}_i^0\rangle = \sum_{i=1}^{n}\langle \mathbf{x}_i^1, \mathbf{y}_i^1\rangle$. However, nothing is guaranteed for the relation between $\langle \mathbf{x}_i^0, \mathbf{y}_i^0\rangle$ and $\langle \mathbf{x}_i^1, \mathbf{y}_i^1\rangle$. For this reason, it happens during the security proof that some inputs $\mathbf{x}_i$ or $\mathbf{y}_i$ that the adversary queries to the encryption or key-generation oracle must be "mixed" and embedded in the responses to prior or subsequent queries. In the adaptive setting, this can lead to the situation that the simulator needs to embed values into decryption keys or ciphertexts before they are actually queried by the adversary. In the MCFE of [AGT21b], this problem is solved by resorting to selective security. In this case, the simulator knows all the inputs from the beginning and can use them whenever necessary. In contrast, we provide a concrete instantiation of the underlying single-input FE scheme based on DPVS. If the simulator gets into a situation where it would have to use inputs that have not yet been queried by the adversary, we make it guess them. Even though this makes the simulator inefficient, the adversary cannot exploit this fact due to information-theoretic properties of the DPVS setting. We emphasize that this technique cannot be applied to black-box instantiations of the inner-product FE scheme because they provide only computational security in general.

On the negative side, the security model of our DMCFE scheme suffers from the restrictions that the adversary can ask challenge ("left-or-right") queries only for one tag. This constraint does not exist in the MCFE of [AGT21b]. The intuitive reason is that the above-mentioned information-theoretic arguments force us to consider queries one-by-one while the security proof in [AGT21b] deals with all queries at the same time in a uniform way. Fortunately, a simple hybrid argument shows that one-challenge and multi-challenge security are equivalent in the weak function-hiding setting. Details can be found in Lemmas 8. Intuitively, in the standard setting (without function-hiding) the proof can be done by a sequence of hybrids over the different public inputs queried to the encryption oracle. However, this approach does not directly generalize to the function-hiding setting. The problem is that now both encryption and key-generation queries depend on the challenge bit $b \in \{0,1\}$. Since ciphertexts and keys can be arbitrarily combined in general, such a sequence of hybrids leads to a situation where an adversary is able to mix ciphertexts that encrypt the left message with keys generated for the right function or vice versa. However, the function-hiding admissibility does not provide any security guarantees in the case of such mixed decryption. Therefore, we cannot change ciphertexts and keys one by one anymore. We solve this problem by means of a standard technique. First, we prove security in the weakly function-hiding setting. This model provides us exactly with the necessary guarantee for mixed decryptions, which allows us to subsequently swap keys and ciphertexts. Afterwards, we apply a standard transformation that turns weakly function-hiding FE schemes for inner products into schemes that enjoy full-fledged function-hiding security. Previous works [LV16, ACF+18] presented the transformation for single-input and multi-input FE schemes. For completeness, we argue the case of DDFE in Lemma 9.

Last but not least, our FH-DMCFE scheme does not require the one key-label restriction gracefully using our core Lemma 12, in the context of swapping multiple families of vectors $(\mathbf{c}_i^{(j)}, \mathbf{d}_i^{(\tilde{j})})_i$, where at a position $i \in [n]$ there might be repetitions of $\mathbf{c}_i^{(j)}$ or $\mathbf{d}_i^{(\tilde{j})}$ (see Section 3.1). The proof heavily relies on information-theoretic properties of DPVS and we refer to Section 4 for general ideas therein.

Challenge 2: Non-Interactive Generation of Secret Keys. To overcome the second challenge, it is natural to look for a way to generate the secret keys of the users without interaction. However, a dynamic set of users (i.e. users can join the system at any time) is only meaningful if the corresponding function class of the scheme supports dynamically chosen support sets. Such a case is not considered in the (D)MCFE setting, as the set of clients $[n]$ is fixed up front. Therefore, it is not enough to set up one instance of the underlying (D)MCFE, but one needs to emulate an independent instance for each support.

Let ID be a set of identities. Recall that in the MCFE of [AGT21b], the secret information about a user $i \in \mathsf{ID}$ consists of a secret key for a single-input function-hiding FE scheme and a secret share $s_i \in \mathbb{Z}_q$. For the former, they equip user i with a key K_i for a family of pseudorandom functions $\{F_K\}_K$. If the user wants to obtain a secret key w.r.t some support $\mathcal{U} \subseteq \mathsf{ID}$, it evaluates $F_{K_i}(\mathcal{U}) \to r_i$ and runs the setup algorithm of the single-input FE scheme with fixed random coins r_i. For the secret share s_i, the authors of [AGT21b] use a technique introduced in [CDSG+20] under the name *decentralized sum (DSUM)*. Basically, this technique uses a clever interleaving of a non-interactive key exchange (NIKE) scheme and a PRF to generate a fresh share s_i for each support $\mathcal{U}$.
Our construction follows exactly this blueprint. That is, we employ a PRF to generate a fresh DPVS instance for each user-support pair $(i,\mathcal{U})$ and a DSUM instance to generate random sharings. Recall from Section 3.1 that our DMCFE scheme embeds secret sharings in both ciphertexts and keys. For this reason, we use the DSUM technique twice, in encryption and key generation generation. To guarantee the independence of the secret sharings, we add some salt to the PRF in both applications.

Challenge 3: Security against Incomplete Queries. To overcome the third challenge, the authors of [AGT21b] make use of a technique called *all-or-nothing encapsulation (AoNE)* that was previously introduced in [CDSG+20]. Roughly, AoNE allows all parties of a (previously fixed) group to encapsulate individual messages, that can *all* be extracted by everyone if and only if all parties of the group have sent their contribution. Otherwise, *no* message is revealed. In the DDFE constructions of [CDSG+20, AGT21b], such an AoNE layer is added on top of both ciphertexts and keys. Intuitively, this approach allows the following reasoning: if an adversary makes encryption queries for all (honest) clients under some tag tag (i.e. the query is "complete"), then the AoNE scheme allows to obtain all ciphertexts, and we can rely on the security of the DDFE scheme that is secure against complete challenges. On the other hand, if the adversary queries only some but not all honest clients (i.e. the query is "incomplete"), then the security of the AoNE scheme guarantees that the adversary does not learn anything about the encapsulated messages. While this construction is well known, previous DDFE constructions prove only selective security, even if the employed AoNE scheme is adaptively secure. Specifically, the constructions of [CDSG+20, AGT21b] already resort to selective security to deal with the second challenge. For simplicity, their security proof then also exploits selective security to handle incomplete queries. Nevertheless, we believe it is important to show that this AoNE layer indeed preserves adaptive security if the underlying DDFE, which is secure against complete challenges, has this property. To our knowledge, the only known conversion with this property has been presented in [ABG19]. The drawback of that construction is that ciphertexts grow quadratically in the number of clients. So, in conclusion, we show that the AoNE technique of [CDSG+20] achieves the same security level as [ABG19], while having ciphertexts of only linear size if an appropriate instantiation for the AoNE scheme is used.
We present our results in form of a generic conversion that turns any one-challenge DDFE scheme secure against complete queries into one that is also secure against incomplete queries. The formal description of the conversion is provided in Appendix B.7. Security is stated in Lemma 16. On an intuitive level, our simulator initially guesses whether or not the encryption queries for the challenge public input k^*_{pub} (or m^*_{pub}) will be complete. If the guess was "complete" and this guess turns out to be correct at the end of the game, then the simulator attacks the underlying DDFE scheme that is assumed to be secure against complete queries. If the guess was "incomplete" and the guess is correct, then the simulator attacks the security of the AoNE scheme. If the guess was incorrect (which happens with probability 1/2), then the simulator aborts with a random bit. In this way, we can upper bound the advantage of a distinguisher between two successive hybrids in terms of the advantages that efficient adversaries can achieve against the underlying AoNE and DDFE schemes. We point out that this conversion crucially relies on the *one-challenge* setting. Due to the guess

on the (in)completeness of the query, we lose a factor $1/2$ in the security proof. Thus, a hybrid argument over a polynomial number of incomplete queries would incur an exponential security loss. Therefore, it is important to add security against incomplete queries in the one-challenge model, while noticing that security against incomplete non-challenge queries follows trivially. Afterwards, one can obtain security against multiple challenges using the sequence of hybrids described above. We think that this corrects a subtle flaw in the proof of [SV23, Theorem 5] where the authors present a similar conversion in the MCFE setting. Nevertheless, by using our workaround via one-challenge security their result remains correct.

4 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. A simple version is presented in Lemma 11, then the full indistinguishability property is stated in its corollary, Lemma 12.

Lemma 11 (Single Secret-Sharing Swapping). *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda), J = J(\lambda), N_i = N_i(\lambda) \in \mathbb{N}$ where $i \in H$ and $H, K, L, J, N_i : \mathbb{N} \to \mathbb{N}$ are polynomials. Let $(\mathbf{B}_i, \mathbf{B}_i^*)_{i\in[H]}$ be two random dual bases of dimension $4N_i + 4$ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$. All basis vectors are kept secret.*
For each $i \in [H]$, consider public vectors $(\mathbf{x}_i, \mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}_{\ell,i}^{(\mathsf{rep})'}, \mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)}, \mathbf{y}_{k,i}^{(\mathsf{rep})})_{k\in[K],\ell\in[L]}^{\mathsf{rep},j\in[J]}$ of length N_i such that $\sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(0,j)}\rangle = \sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(1,j)}\rangle$ and $\langle \mathbf{x}_i, \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}\rangle$ is a constant for all $j \in [J], i \in [H]$. Let $(R, R_k)_{j\in[J],k\in[K]}$ be some public scalars. Given $r, r_\ell, \rho_i, \rho_{\ell,i}^{(\mathsf{rep})}, \pi_i^{(j)}, \pi_{k,i}^{(\mathsf{rep})}, \sigma_i, \sigma_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ 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:

$$D_0 := \left\{ \begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ \quad (\mathbf{u}_i = (\boxed{\mathbf{x}_i},\ \boxed{0^{N_i}},\ r,\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H]} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\};$$

$$D_1 := \left\{ \begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ \quad (\mathbf{u}_i = (\boxed{0^{N_i}},\ \boxed{\mathbf{x}_i},\ r,\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H]} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\}.$$

More specifically, we have:

$$\left| \Pr_{\mathsf{samp}\sim D_0}[\mathcal{A}(\mathsf{samp}) \to 1] - \Pr_{\mathsf{samp}\sim D_1}[\mathcal{A}(\mathsf{samp}) \to 1] \right| \leq (2N_i + 8) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

for any ppt $\mathcal{A}$ *with fixed* $\left(\mathbf{x}_i, (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}_{\ell,i}^{(\mathsf{rep})'})_{\ell=1}^L, R, \mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)}, (\mathbf{y}_{k,i}^{(\mathsf{rep})}, R_k)_{k=1}^K\right)_{i\in[H]}^{j\in[J]}$.

The proof of Lemma 11 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) $\forall\, j \in [J] : \sum_{i=1}^{H} \tau_i = 0$

$$
\begin{array}{lcccccccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& \mathbf{x}_i & 0^{N_i} & \boxed{r} & 0 & \rho_i & 0^{N_i} & 0^{N_i} & \boxed{r'} &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \boxed{\sigma_i} & \pi_i^{(j)} & 0 & 0^{N_i} & 0^{N_i} & \boxed{\tau_i} &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_2: (Formal Duplication from coordinates $[1, N_i], [N_i + 1, 2N_i]$ to $[2N_i + 4, 3N_i + 3], [3N_i + 4, 4N_i + 3]$ in $\mathbf{B}_i^*$)

$$
\begin{array}{lcccccccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& \mathbf{x}_i & 0^{N_i} & r & 0 & \rho_i & 0^{N_i} & 0^{N_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \boxed{\mathbf{y}_i^{(1,j)}} & \boxed{\mathbf{y}_i^{(0,j)}} & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_3: (Computational Swapping between $[1, N_i]$ and $[2N_i + 4, 3N_i + 3]$ in $\mathbf{u}_i$ using $(2N_i + 3)$-randomness in $\mathbf{B}_i$)

$$
\begin{array}{lcccccccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& \boxed{0^{N_i}} & 0^{N_i} & r & 0 & \boxed{\rho_i} & \boxed{\mathbf{x}_i} & 0^{N_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Inside a *complexity leveraging* argument:

Game G_4: (Formal Quotient on coordinates $[2N_i + 4, 4N_i + 3]$ in $\mathbf{B}_i$)

$$
\begin{array}{lcccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \cdots & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& \cdots & 1^{N_i} & 0^{N_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \cdots & (\mathbf{x}_i[m]\mathbf{y}_i^{(1,j)}[m])_m & (\mathbf{x}_i[m]\mathbf{y}_i^{(0,j)}[m])_m & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \cdots & (\mathbf{x}_i[m]\mathbf{y}_{k,i}^{(\mathsf{rep})}[m])_m & (\mathbf{x}_i[m]\mathbf{y}_{k,i}^{(\mathsf{rep})}[m])_m & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_5: $\Delta\mathbf{y}_i^{(j)} \coloneqq \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}, \tilde{\tau}_i \coloneqq \tau_i + \frac{1}{r'}\langle \mathbf{x}_i, \Delta\mathbf{y}_i^{(j)} \rangle$ (Formal Swapping)

$$
\begin{array}{lcccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \cdots & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& \cdots & \boxed{0^{N_i}} & \boxed{1^{N_i}} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \cdots & (\mathbf{x}_i[m]\mathbf{y}_i^{(1,j)}[m])_m & (\mathbf{x}_i[m]\mathbf{y}_i^{(0,j)}[m])_m & \boxed{\tilde{\tau}_i} &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \cdots & (\mathbf{x}_i[m]\mathbf{y}_{k,i}^{(\mathsf{rep})}[m])_m & (\mathbf{x}_i[m]\mathbf{y}_{k,i}^{(\mathsf{rep})}[m])_m & \boxed{0} &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_6: (Formal Quotient on coordinates $[2N_i + 4, 4N_i + 3]$ in $\mathbf{B}_i$)

$$
\begin{array}{lcccccccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& 0^{N_i} & 0^{N_i} & r & 0 & \rho_i & \boxed{0^{N_i}} & \boxed{\mathbf{x}_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \boxed{\mathbf{y}_i^{(1,j)}} & \boxed{\mathbf{y}_i^{(0,j)}} & \tilde{\tau}_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_7: (Computational Swapping between $[1, N_i]$ and $[3N_i + 4, 4N_i + 3]$ in $\mathbf{u}_i$ using $(2N_i + 3)$-randomness in $\mathbf{B}_i$)

$$
\begin{array}{lcccccccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& 0^{N_i} & \boxed{\mathbf{x}_i} & r & 0 & \rho_i & 0^{N_i} & \boxed{0^{N_i}} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \tilde{\tau}_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_8: Undo G_2, G_1 (Cleaning) – Vectors sampled according to D_1.

$$
\begin{array}{lcccccccccl}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& 0^{N_i} & \mathbf{x}_i & r & 0 & \boxed{\rho_i} & 0^{N_i} & \boxed{0^{N_i}} & \boxed{0} &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \boxed{0^{N_i}} & \boxed{0^{N_i}} & \boxed{0} &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \boxed{0^{N_i}} & \boxed{0^{N_i}} & 0 &)_{\mathbf{B}_i^*}
\end{array}
$$

Fig. 2: Games for proving Lemma 11

Proof (Main ideas). The sequence of games is given in Figure 2. We explain the main steps in our proof as follows. We start from the game where the sample given to the adversary $\mathcal{A}$ follows D_0 and the changes on vectors throughout the games are put in boxes. Our first step is to exploit the fact that $r \xleftarrow{\$} \mathbb{Z}_q$ is a uniformly random value and for each $j \in [J]$ all the secret shares σ_i in $\mathbf{v}_i^{(j)}$ sum to a known constant 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 $\mathbf{u}_i[4N_i+4]$ as well as a random secret sharing of 0, common for $j \in [J]$, namely $(\tau_i)_{i=1}^H$, in $(\mathbf{v}_i^{(j)}[4N_i+4])_{i=1}^H$.

$$\begin{array}{lccccccccc}
\mathbf{u}_i = (& \mathbf{x}_i & 0^{N_i} & \boxed{r} & 0 & \rho_i & 0^{N_i} & 0^{N_i} & \boxed{r'} &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \boxed{\sigma_i} & \pi_i^{(j)} & 0 & 0^{N_i} & 0^{N_i} & \boxed{\tau_i} &)_{\mathbf{B}_i^*}
\end{array}$$

We need a DSDH (see Appendix A.4) instance $[\![(a,b,c)]\!]_1$ in $\mathbb{G}_1$ where $c-ab$ is either 0 or 1 to introduce the non-zero value $r' \xleftarrow{\$} \mathbb{Z}_q^*$, following the subspace indistinguishability. The DSDH instance does not depend on i nor j but because the basis change is computational, we can use $[\![(a,b,c)]\!]_1$ to compose linear combinations of vectors with coefficients depending on i and j. Those that are written in $(\mathbf{H}_i, \mathbf{H}_i^*)$ will be changed accordingly into new writings in $(\mathbf{B}_i, \mathbf{B}_i^*)$, making $(c-ab)r'$ appear in $\mathbf{u}_i[4N_i+4]$ and from which we can conclude the indistinguishability using DSDH, whose hardness is related to DDH. It is indispensible that we make this basis change only for honest i, which are known statically, due to the fact that we need all basis vectors to be kept secret. Otherwise by issuing a corruption the adversary will detect the simulation because we cannot simulate $\mathbf{B}_i^*$ without $[\![a]\!]_2$. The randomness br' is taken from r at coordinate $(2N_i+1)$. We stress that this type of computational changes under SXDH is important for our proof, especially when we aim at changing only *some* vectors, *e.g.* in the above we do not want to touch $\mathbf{u}_{\ell,i}^{(\mathsf{rep})}$ nor $\mathbf{v}_{k,i}^{(\mathsf{rep})}$ that can be left out of the computation involving $(c-ab)r'$. Similarly a DSDH instance $[\![(a,b,c)]\!]_2$ in $\mathbb{G}_2$ is required when introducing $(\tau_i)_{i=1}^H$. More details can be found in the transition $\mathsf{G}_0 \to \mathsf{G}_1$.

Another type of basis changes in DPVS that we will need is changing bases formally, *i.e.* writing the $\mathbf{u}$-vectors and $\mathbf{v}$-vectors in $(\mathbf{H}_i, \mathbf{H}_i^*)$ according to the original game and the basis change will modify *all* vectors into $(\mathbf{B}_i, \mathbf{B}_i^*)$, which corresponds to the next game. This type of basis changes modifies *all* vectors and keeps the adversary's views over two games perfectly identical. For instance, after G_1, we perform a formal duplication to go to G_2 in which we duplicate coordinates $[1,N_i], [N_i+1, 2N_i]$ to $[2N_i+4, 3N_i+3], [3N_i+4, 4N_i+3]$ in vectors $\mathbf{v}_i^{(j)}, \mathbf{v}_{k,i}^{(\mathsf{rep})}$ for all $i \in [H], k \in [K], j \in [J]$.

$$\begin{array}{lccccccccc}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& \mathbf{x}_i & 0^{N_i} & r & 0 & \rho_i & 0^{N_i} & 0^{N_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \boxed{\mathbf{y}_i^{(1,j)}} & \boxed{\mathbf{y}_i^{(0,j)}} & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & 0 &)_{\mathbf{B}_i^*}
\end{array}$$

The duplication is done for *all* vectors $\mathbf{v}_i^{(j)}, \mathbf{v}_{k,i}^{(\mathsf{rep})}$ also across all repetitions $\mathsf{rep} \in [J]$. On a more technical detail, this formal basis change will affect *all* vectors $\mathbf{u}_{\ell,i}^{(\mathsf{rep})}, \mathbf{u}_i$ as well, also across all repetitions $\mathsf{rep} \in [J]$. Roughly speaking, by the duality of $(\mathbf{B}_i, \mathbf{B}_i^*)$, this basis change will incur "moving" coordinates $[2N_i+4, 3N_i+3], [3N_i+4, 4N_i+3]$ to $[1, N_i], [N_i+1, 2N_i]$ in the $\mathbf{u}$-vectors. In this simple $\mathsf{G}_1 \to \mathsf{G}_2$ the moved coordinates contain 0 and that poses no problems. All calculation can be found in the full proof of Appendix B.3.

After G_2, we perform a computational basis change under SXDH in order to swap between $[1, N_i]$ and $[2N_i+4, 3N_i+3]$ in $\mathbf{u}_i$. We again use a DSDH instance in $\mathbb{G}_1$ to perform this computational basis change, where the randomness is taken from ρ_i at coordinate $2N_i+3$ in $\mathbf{u}_i$.

$$\begin{array}{lcccccccccc}
\mathbf{u}_i = (& \boxed{0^{N_i}} & 0^{N_i} & r & 0 & \boxed{\rho_i} & \boxed{\mathbf{x}_i} & 0^{N_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \tau_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & 0 &)_{\mathbf{B}_i^*}
\end{array} .$$

We remark that this change preserves the products $\mathbf{u}_i \times \mathbf{v}_i^{(j)}$ and $\mathbf{u}_i \times \mathbf{v}_{k,i}^{(\mathsf{rep})}$ for all $k \in [K], j \in [J]$ necessarily for the indistinguishability. Moreover, the computational basis change allows us to target only the vectors $(\mathbf{u}_i)_{i\in[H]}$ while maintaining $\mathbf{u}_{\ell,i}^{(\mathsf{rep})}$ for $\ell \in [L], i \in [H]$ intact.

Once we arrive at G_3, we have the required ingredients for the core of our proof. As mentioned earlier, a formal basis keeps the adversary's views over two games perfectly identical and this enables us to use a *complexity leveraging* argument in the following. The goal of complexity leveraging is to prove that the adversary's views over two hybrids are perfectly identical, i.e. the difference in advantages for winning the hybrids is 0 under efficient simulation. The security loss is 0 thanks to a completely formal argument on top of two perfectly identical variants of the hybrids, for which the adversary's views are already identical. The links between underlying variants for which we need identical views will be done using only formal basis changes. The possibility to perform these formal changes accentuates the information-theoretic properties of DPVS. The main challenge is to handle the situation that under those basis changes *all* vectors will be modified.

After G_3, we want to perform some sort of swapping between coordinates $[2N_i + 4, 3N_i + 3]$ and $[3N_i + 4, 4N_i + 3]$ of $\mathbf{u}_i$ and reach G_6 whose vectors are:

$$\begin{array}{lcccccccccc}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (& \mathbf{x}_{\ell,i}^{(\mathsf{rep})} & \mathbf{x}_{\ell,i}^{(\mathsf{rep})'} & r_\ell & 0 & \rho_{\ell,i}^{(\mathsf{rep})} & 0^{N_i} & 0^{N_i} & 0 &)_{\mathbf{B}_i} \\
\mathbf{u}_i = (& 0^{N_i} & 0^{N_i} & r & 0 & \rho_i & \boxed{0^{N_i}} & \boxed{\mathbf{x}_i} & r' &)_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} = (& \mathbf{y}_i^{(1,j)} & \mathbf{y}_i^{(0,j)} & \sigma_i & \pi_i^{(j)} & 0 & \boxed{\mathbf{y}_i^{(1,j)}} & \boxed{\mathbf{y}_i^{(0,j)}} & \tilde{\tau}_i &)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} = (& \mathbf{y}_{k,i}^{(\mathsf{rep})} & \mathbf{y}_{k,i}^{(\mathsf{rep})} & \sigma_{k,i} & \pi_{k,i}^{(\mathsf{rep})} & 0 & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})}} & 0 &)_{\mathbf{B}_i^*}
\end{array} .$$

The complexity leveraging will be applied to the *selective* versions $\mathsf{G}_3^* \to \mathsf{G}_4^* \to \mathsf{G}_5^* \to \mathsf{G}_6^*$ and only formal basis changes will be used in between. In these selective versions the simulator guesses the values $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N_i]}^{j\in[J]}$ and the hybrids are conditioned on a "good" event that happens with fixed probability. This leads to an identical adversary's view:

$$\Pr[\mathsf{G}_3^* = 1] = \Pr[\mathsf{G}_4^* = 1] = \Pr[\mathsf{G}_5^* = 1] = \Pr[\mathsf{G}_6^* = 1] \ . \tag{5}$$

We briefly highlight the games' ideas below:

- In $\mathsf{G}_3^* \to \mathsf{G}_4^*$ a formal basis change is applied to do a quotient by $\mathbf{x}_i[k]$ for $k \in [n_i]$ over all coordinates $[2N_i + 4, 3N_i + 3]$ as well as $[3N_i + 4, 4N_i + 3]$ of $\mathbf{v}$-vectors. There are some technicalities when defining the basis matrices to ignore the quotient when $\mathbf{x}_i[k] = 0$ and we refer to equation (9) in the proof for more details.
- Then, in $\mathsf{G}_4^* \to \mathsf{G}_5^*$, we define a formal basis change that uses the *fixed* randomness $r' \in \mathbb{Z}_q^*$ in $\mathbf{u}_i[4N_i + 4]$ to switch 1 to 0 at coordinates $[2N_i + 4, 3N_i + 3]$ and 0 to 1 at coordinates $[3N_i + 4, 4N_i + 3]$ of all $\mathbf{u}_i$. The matrix definition is given in equation (10). We note that unlike $\mathbf{u}_i$, $\mathbf{u}_{\ell,i}^{(\mathsf{rep})}[4N_i + 4] = 0$ hence these vectors stay invariant under this change.
  - Dually, all $\mathbf{v}$-vectors will be altered such that the *difference* of entries at two coordinates $(2N_i + 3 + k, 3N_i + 3 + k)$ for all $k \in [N_i]$ will be "moved" to coordinates $(4N_i + 4)$. For $\mathbf{v}_{k,i}^{(\mathsf{rep})}$ we have $\mathbf{v}_{k,i}^{(\mathsf{rep})}[2N_i + 3 + k] = \mathbf{v}_{k,i}^{(\mathsf{rep})}[3N_i + 3 + k]$ and those differences are all 0, no matter which repetition rep. The real challenge comes from the fact that we have *one* set of $(\mathbf{u}_i)_{i=1}^H$ but *multiple* sets of $(\mathbf{v}_i^{(j)})_{i=1}^H$ for each $j \in [J]$, where $\mathbf{v}_i^{(j)}[2N_i + 3 + k] - \mathbf{v}_i^{(j)}[3N_i + 3 + k] = \mathbf{x}_i \Delta \mathbf{y}_i^{(j)}[k] \coloneqq \mathbf{x}_i(\mathbf{y}_i^{(1,j)}[k] - \mathbf{y}_i^{(0,j)}[k])$ that could be non-zero.

– It is at this point that we need to use the secret sharing of 0 for each $j \in [J]$, *i.e.* $(\tau_i)_{i=1}^H$ using the hypothesis that $\sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(0,j)} \rangle = \sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(1,j)} \rangle$ for all $j \in [J]$. More specifically, adding a fixed multiple of $\langle \mathbf{x}_i, \Delta\mathbf{y}_i^{(j)} \rangle$ to τ_i, which is constant for whatever j, still keeps it a sharing of 0 over all $i \in [H]$ and the new $\tilde{\tau}_i$ still does not depend on j. This "fixed multiple" depends only on $r' \xleftarrow{\$} \mathbb{Z}_q^*$ and thus preserves the distribution of $(\tau_i)_{i=1}^H$.

- Finally, in $\mathsf{G}_5^* \to \mathsf{G}_6^*$ we redo the quotient, still being in the selective variants conditioned on the "good" event.

The probability calculation (see footnote 7) of the complexity leveraging makes use of the fact that the "good" event happens with a fixed probability as well as property (5), leading to $\Pr[\mathsf{G}_3 = 1] = \Pr[\mathsf{G}_4 = 1] = \Pr[\mathsf{G}_5 = 1] = \Pr[\mathsf{G}_6 = 1]$. Coming out of the complexity-leveraging argument, the very last step consists in swapping $\mathbf{x}_i$ from coordinates $[3N_i + 4, 4N_i + 3]$ back to $[1, N_i]$ (see $\mathsf{G}_6 \to \mathsf{G}_7$) and some cleaning in order to make the vectors follow D_1 (see $\mathsf{G}_7 \to \mathsf{G}_8$). □

Lemma 12 (Secret Sharing Swapping with Repetitions). *Under the same hypotheses of Lemma 11, additionally with repetitions* $\mathbf{u}_i^{(\tilde{j})}$ *for* $\tilde{j} \in [J]$ *so that*

$$\sum_{i=1}^{H} \langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(0,j)} \rangle = \sum_{i=1}^{H} \langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)} \rangle \tag{6}$$

and for each $i \in [H]$, $\langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)} \rangle = c_i$ *being a constant for all* $\tilde{j}, j \in [J]$, *the following distributions are computationally indistinguishable under the* SXDH *assumption in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathbf{t}, g_1, g_2, g_\mathbf{t}, \mathbf{e}, q)$*:*

$$D_0 := \left\{ \begin{array}{l} \left(\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\right)_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ \left(\mathbf{u}_i^{(\tilde{j})} = (\mathbf{x}_i^{(\tilde{j})},\ 0^{N_i},\ r,\ 0,\ \rho_i^{(\tilde{j})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\right)_{i\in[H]}^{\tilde{j}\in[J]} \\ \left(\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H]}^{j\in[J]} \\ \left(\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\};$$

$$D_1 := \left\{ \begin{array}{l} \left(\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\right)_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ \left(\mathbf{u}_i^{(\tilde{j})} = (0^{N_i},\ \mathbf{x}_i^{(\tilde{j})},\ r,\ 0,\ \rho_i^{(\tilde{j})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\right)_{i\in[H]}^{\tilde{j}\in[J]} \\ \left(\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H]}^{j\in[J]} \\ \left(\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\right)_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\}.$$

More specifically, we have:

$$\left| \Pr_{\mathsf{samp}\sim D_0}[\mathcal{A}(\mathsf{samp}) \to 1] - \Pr_{\mathsf{samp}\sim D_1}[\mathcal{A}(\mathsf{samp}) \to 1] \right| \leq (2N_i + 8) \cdot J \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

for any ppt $\mathcal{A}$ *with fixed* $\left(\mathbf{x}_i, (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}_{\ell,i}^{(\mathsf{rep})'})_{\ell=1}^L, R, \mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)}, (\mathbf{y}_{k,i}^{(\mathsf{rep})}, R_k)_{k=1}^K\right)_{i\in[H]}^{j\in[J]}$.

The proof is done via a sequence of hybrids over the repetitions $\tilde{j}$ where we separate the repetition $\mathbf{u}_i^{(\tilde{j})}$ into isolated coordinates then apply Lemma 11. A full proof can be found in Appendix B.4.

5 Towards FH-DDFE: A Weakly Function-Hiding DMCFE for Inner Products

This section presents an FH-DMCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the function class $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ that is defined in Definition 5. We work in the prime-order bilinear group setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$ and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}$ are all written additively. We employ two full-domain hash functions $\mathsf{H}_1 \colon \mathsf{Tag} \to \mathbb{G}_1$ and $\mathsf{H}_2 : \mathsf{Tag} \to \mathbb{G}_2$. The details of $\mathcal{E}$ go as follows:

$\mathsf{Setup}(1^\lambda)$**:** Run $\mathsf{DPVSGen}(\mathbb{G}, 1^{5N_i+5}; r_i \xleftarrow{\$} \{0,1\}^*)$ for $i \in [n]$ to obtain n pairs of dual orthogonal bases $(\mathbf{B}_i, \mathbf{B}^*_i)$ for $i \in [n]$ of dimensions $5N_i + 5$ for each $(\mathbf{B}_i, \mathbf{B}^*_i)$. 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:

$$\begin{aligned}\mathsf{sk}_i &:= (\mathbf{b}^*_{i,1}, \ldots, \mathbf{b}^*_{i,N_i},\ \tilde{s}_i B^*_{i,N_i+1} + B^*_{i,3N_i+2},\ \mathbf{b}^*_{i,3N_i+3})\\ \mathsf{ek}_i &:= (\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,N_i},\ \mathbf{b}_{i,3N_i+4},\ B_{i,N_i+1} + \tilde{t}_i B_{i,3N_i+2})\end{aligned}$$

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

$\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, \mathbf{y}_i)$**:** Parse $\mathsf{sk}_i = (\mathbf{b}^*_{i,1}, \ldots, \mathbf{b}^*_{i,N_i}, \tilde{s}_i B^*_{i,N_i+1} + B^*_{i,3N_i+2}, \mathbf{b}^*_{i,3N_i+4})$. Compute $\mathsf{H}_2(\mathsf{tag\text{-}f}) \to [\![\mu]\!]_2 \in \mathbb{G}_2$ and sample $\pi_i \xleftarrow{\$} \mathbb{Z}_q$. Compute and output

$$\begin{aligned}\mathbf{d}_i &= \sum_{k=1}^{N_i} \mathbf{y}_i[k]\mathbf{b}^*_{i,k} + (\tilde{s}_i B^*_{i,N_i+1} + B^*_{i,3N_i+2}) \cdot [\![\mu]\!]_2 + \pi_i \mathbf{b}^*_{i,3N_i+3}\\ &= (\mathbf{y}_i,\ \tilde{s}_i\mu,\ 0^{N_i},\ 0^{N_i},\ \mu,\ \pi_i,\ 0,\ 0^{2N_i+1})_{\mathbf{B}^*_i}\ .\end{aligned}$$

$\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$**:** Parse $\mathsf{ek}_i = (\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,N_i},\ \mathbf{b}_{i,3N_i+4},\ B_{i,N_i+1} + \tilde{t}_i B_{i,3N_i+2})$. Compute $\mathsf{H}_1(\mathsf{tag}) \to [\![\omega]\!]_1 \in \mathbb{G}_1$ and sample a random scalar $\rho_i \xleftarrow{\$} \mathbb{Z}_q$. Finally, compute and output

$$\begin{aligned}\mathbf{c}_i &= \sum_{k=1}^{N_i} \mathbf{x}_i[k]\mathbf{b}_{i,1} + \rho_i \mathbf{b}_{i,3N_i+4} + (B_{i,N_i+1} + \tilde{t}_i B_{i,3N_i+2}) \cdot [\![\omega]\!]_1\\ &= (\mathbf{x}_i,\ \omega,\ 0^{N_i},\ 0^{N_i},\ \tilde{t}_i\omega,\ 0,\ \rho_i,\ 0^{2N_i+1})_{\mathbf{B}_i}\ .\end{aligned}$$

$\mathsf{Dec}(\mathbf{d}, \mathbf{c})$**:** 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:

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

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

Security. Theorem 13 states that the scheme is *weakly function-hiding* under *static corruption* and a *single adaptive challenge*, allowing *repetitions*. We recall that for the moment the security is proven under the *complete* queries constraints (Condition 4 in Definition 18). This constraint does not affect the resulting security of our FH-DDFE in Section 6 that uses the current FH-DMCFE. We can apply a generic conversion to lift it from the final FH-DDFE and still preserve adaptive security.

Theorem 13. *The DMCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* adaptively one-challenge weakly 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 maximum number of distinct tags for ciphertexts to* $\mathcal{O}\mathsf{Enc}$, q_k *denote the maximum number of distinct tags for keys to* $\mathcal{O}\mathsf{KeyGen}$, J *denote the maximum number of repetitive challenge queries for possibly different messages, and* $\tilde{J}$ *denote the maximum number of repetitive challenge queries for possibly different keys. Then, for any* ppt *adversary* $\mathcal{A}$ *against* $\mathcal{E}$, *we have the following bound:*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \leq \left(\left((q_e+1)J + (q_k+1)\tilde{J}\right)\cdot(2N_i+8) + q_k + q_e + 8N_i + 2\right)\cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

In Figure 5, we present the sequence of games used to prove Theorem 13, where we start from G_0 corresponding to $\mathbf{Exp}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, where $b \xleftarrow{\$} \{0,1\}$ is the random challenge bit, until arriving at G_6 that is totally independent of b. We recall that in our proof, we exploit fully the concrete interpretation of *admissible* adversaries against FH-DMCFE for $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ and refer to paragraph *Concrete interpretation of the function-hiding admissibility* in Section 3.1 for more details. The full proof can be found in Appendix B.5, whose ideas are presented in Section 3.1.

6 Function-Hiding DDFE for Inner Products: Adaptive Security under Static Corruption

This section presents a function-hiding DDFE scheme for the function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ defined in Definition 6. As a first step, we show how to upgrade the one-challenge weakly function-hiding DMCFE scheme from Section 5 to a DDFE scheme for the same security level in a semi black-box manner. Afterwards, we remove several constraints in the security model inherited from the DMCFE scheme. Specifically, we show how to handle multiple challenges, incomplete queries and lift to full-fledged function-hiding, thus obtaining a FH-DDFE scheme with adaptive security under static corruption.
For the conversion from DMCFE to DDFE, we employ a slight modification of the DMCFE scheme from Section 5 where encryption and secret keys contain complete information about their respective DPVS. More precisely, let $\mathcal{E}' = (\mathcal{E}'.\mathsf{Setup}, \mathcal{E}'.\mathsf{DKeyGen}, \mathcal{E}'.\mathsf{Enc}, \mathcal{E}'.\mathsf{Dec})$ be the DMCFE scheme of Section 5 except that the Setup' algorithm is defined as follows.

$\mathsf{Setup}(1^\lambda)$**:** Run $(B_i, B_i^*) \leftarrow \mathsf{DPVSGen}(\mathbb{G}, 1^{5N_i+5})$ for $i \in [n]$ and sample $(\tilde{s}_i)_{i\in[n]}, (\tilde{t}_i)_{i\in[n]} \xleftarrow{\$} \mathbb{Z}_q^n$ such that $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$. Then, output the encryption and secret keys $\mathsf{ek}_i := (B_i, \tilde{t}_i), \mathsf{sk}_i := (B_i^*, \tilde{s}_i)$ for all $i \in [n]$.

The following lemma states that $\mathcal{E}'$ preserves the security level of the original DMCFE scheme in Section 5. Given the proof of Theorem 13, Lemma 14 is an immediate consequence of the modular design of our scheme (i.e. different clients use independent DPVS instances) and the fact that we consider only static corruption.

Lemma 14. *The DMCFE scheme* $\mathcal{E}'$ *is adaptively one-challenge weakly function-hiding secure under static corruption in the ROM if the* SXDH *assumption holds for* $(\mathbb{G}_1, \mathbb{G}_2)$. *Moreover, the security proof does not rely on the randomness of shares* $\tilde{s}_i$ *and* $\tilde{t}_i$ *for* $i \in \mathcal{C}$.

Additionally, our DDFE scheme $\mathcal{E}$ employs a NIKE scheme $\mathcal{N} = (\mathcal{N}.\mathsf{Setup}, \mathcal{N}.\mathsf{SharedKey})$ and two families of pseudorandom functions denoted $\{F_K\}_{K\in\mathcal{K}}$ and $\{F'_K\}_{K\in\mathcal{K}'}$. The details of $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ go as follows:

$\mathsf{GSetup}(1^\lambda)$**:** On input the security parameter 1^λ, choose a prime-order bilinear group setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$, run $\mathcal{N}.\mathsf{pp} \leftarrow \mathcal{N}.\mathsf{Setup}(1^\lambda)$ and return $\mathsf{pp} = (\mathbb{G}, \mathcal{N}.\mathsf{pp})$

$\mathsf{LSetup}(\mathsf{pp}, i)$**:** On input the public parameters pp and a user $i \in \mathsf{ID}$, sample $K_i \xleftarrow{\$} \mathcal{K}$, generate $(\mathcal{N}.\mathsf{sk}_i, \mathcal{N}.\mathsf{pk}_i) \leftarrow \mathcal{N}.\mathsf{KeyGen}(\mathcal{N}.\mathsf{pp})$ and return the key pair $(\mathsf{sk}_i := (\mathcal{N}.\mathsf{sk}_i, K_i), \mathsf{pk}_i := \mathcal{N}.\mathsf{pk}_i)$.

$\mathsf{KeyGen}(\mathsf{sk}_i, k)$**:** On input a secret key $\mathsf{sk}_i = (\mathcal{N}.\mathsf{sk}_i, K_i)$ and $k = (\mathbf{y}_i, (\mathcal{U}_K, \mathsf{tag\text{-}f}))$ such that $i \in \mathcal{U}_K$, compute:

$$\begin{aligned} r_i &= F_{K_i}(\mathcal{U}_K) \\ (B_i, B_i^*) &\leftarrow \mathsf{DPVSGen}(\mathbb{G}, 1^{5N_i+5}; r_i) \\ K'_{i,j} &\leftarrow \mathcal{N}.\mathsf{SharedKey}(\mathcal{N}.\mathsf{sk}_i, \mathcal{N}.\mathsf{pk}_j) \\ \tilde{s}_i &= \sum_{j\in\mathcal{U}_K\setminus\{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_K \parallel \text{“key”}) \\ \mathbf{d}_i &= \mathcal{E}'.\mathsf{DKeyGen}(\mathsf{sk}_i = (B_i^*, \tilde{s}_i), \mathsf{tag\text{-}f}, \mathbf{y}_i) \end{aligned}$$

Finally, return $\mathsf{dk}_i := \mathbf{d}_i$.

$\mathsf{Enc}(\mathsf{sk}_i, m)$**:** On input a secret key sk_i and $m = (\mathbf{x}_i, (\mathcal{U}_M, \mathsf{tag\text{-}f}))$ such that $i \in \mathcal{U}_M$, compute:

$$\begin{aligned} r_i &= F_{K_i}(\mathcal{U}_M) \\ (B_i, B_i^*) &\leftarrow \mathsf{DPVSGen}(\mathbb{G}, 1^{5N_i+5}; r_i) \\ K'_{i,j} &\leftarrow \mathcal{N}.\mathsf{SharedKey}(\mathcal{N}.\mathsf{sk}_i, \mathcal{N}.\mathsf{pk}_j) \\ \tilde{t}_i &= \sum_{j\in\mathcal{U}_M\setminus\{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_M \parallel \text{“ct”}) \\ \mathbf{c}_i &\leftarrow \mathcal{E}'.\mathsf{Enc}(\mathsf{ek}_i = (B_i, \tilde{t}_i), \mathsf{tag\text{-}f}, \mathbf{x}_i) \end{aligned}$$

Finally, return $\mathsf{ct}_i := \mathbf{c}_i$.

$\mathsf{Dec}((\mathsf{sk}_i)_{i\in\mathcal{U}_K}, (\mathsf{ct}_i)_{i\in\mathcal{U}_M})$**:** On input a list of decryption keys $(\mathsf{dk}_i = \mathbf{d}_i)_{i\in\mathcal{U}_K}$ and a list of ciphertexts $(\mathsf{ct}_i = \mathbf{c}_i)_{i\in\mathcal{U}_M}$, if $\mathcal{U}_K \neq \mathcal{U}_M$ abort with failure, otherwise compute and return $\mathsf{out} \leftarrow \mathcal{E}'.\mathsf{Dec}((\mathbf{d}_i)_{i\in\mathcal{U}_K}, (\mathbf{c}_i)_{i\in\mathcal{U}_M})$.

Correctness. Using the correctness of the NIKE scheme $\mathcal{N}$ that gives $K'_{i,j} = K'_{j,i}$ for all $i, j \in \mathcal{U}_K$, we have

$$\begin{aligned} \sum_{i\in\mathcal{U}_K} \tilde{s}_i &= \sum_{i\in\mathcal{U}_K} \sum_{j\in\mathcal{U}_K\setminus\{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_K \parallel \text{“key”}) = 0 \\ \sum_{i\in\mathcal{U}_K} \tilde{t}_i &= \sum_{i\in\mathcal{U}_K} \sum_{j\in\mathcal{U}_K\setminus\{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_K \parallel \text{“ct”}) = 0 \ . \end{aligned}$$

Thus, $(\tilde{s}_i)_{i\in\mathcal{U}_K}$ and $(\tilde{t}_i)_{i\in\mathcal{U}_K}$ are secret sharings of 0, and correctness of $\mathcal{E}$ follows from that of $\mathcal{E}'$.

Security. We show security in the following theorem.

Theorem 15. *If $\mathcal{N}$ is IND-secure, $\{F_K\}_{K\in\mathcal{K}}$ and $\{F'_{K'}\}_{K'\in\mathcal{K}'}$ are families of pseudorandom functions and the* SXDH *assumption holds in $(\mathbb{G}_1, \mathbb{G}_2)$, then $\mathcal{E}$ is adaptively one-challenge weakly function-hiding secure under static corruption against complete queries in the ROM.*

More precisely, let q_h be the maximum number of queries to the oracle $\mathcal{O}\mathsf{HonestGen}$ and let q_u be an upper bound on the number of distinct sets $\mathcal{U} \subseteq \mathsf{ID}$ that occur in an encryption or key-generation query. Then, for any ppt *adversary $\mathcal{A}$, there exist* ppt *algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_4$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda) \leq\ & q_h \cdot \mathbf{Adv}^{\mathsf{prf}}_{\{F_K\},\mathcal{B}_1}(1^\lambda) + q_h^2 \cdot \mathbf{Adv}^{\mathsf{nike}}_{\mathcal{N},\mathcal{B}_2}(1^\lambda) \\ & + q_h^2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\{F'_{K'}\},\mathcal{B}_3}(1^\lambda) + q_u \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E}',\mathcal{F}^{\mathsf{ip}},\mathcal{B}_4}(1^\lambda)\end{aligned}$$

The proof of Theorem 15 can be found in Appendix B.6.

Security against multiple challenges and incomplete queries. By adding a layer of AoNE encryption on top of both ciphertexts and keys, we can make $\mathcal{E}$ secure against incomplete queries. While this technique is well known from [CDSG+20, AGT21b], we provide a new proof showing that the conversion preserves adaptive security if both $\mathcal{E}$ and the AoNE scheme are adaptively secure.

Lemma 16. *Assume there exist (1) a one-challenge, weakly function-hiding DDFE scheme $\mathcal{E}^{\mathsf{pos}}$ for the function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that is secure against complete queries, and (2) an AoNE scheme $\mathcal{E}^{\mathsf{aone}}$ whose message space contains the ciphertext space of $\mathcal{E}^{\mathsf{pos}}$. Then there exists a one-challenge, weakly function-hiding DDFE scheme $\mathcal{E}$ for the same function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that is even secure against incomplete queries.*
More precisely, for any ppt *adversary $\mathcal{A}$, there exist* ppt *algorithms $\mathcal{B}$ and $\mathcal{B}'$ such that*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda) \leq 6 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}xxx\text{-}wfh}}_{\mathcal{E}^{\mathsf{pos}},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{B}}(1^\lambda) + 6 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}^{\mathsf{aone}},\mathcal{F}^{\mathsf{aone}},\mathcal{B}'}(1^\lambda) \ ,$$

where $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}\}$.

Details are given in Appendix B.7. We stress that this conversion crucially relies on the one-challenge setting. Eventually, we apply Lemmas 8 and 9 to obtain a DDFE scheme that is secure against multiple challenges in the fully function-hiding setting.

Acknowledgments

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

References

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

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

ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Heidelberg, April 2019.

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

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

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

AGT21a. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 208–238, Virtual Event, August 2021. Springer, Heidelberg.

AGT21b. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 224–255. Springer, Heidelberg, November 2021.

AGT22. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption: Stronger security, broader functionality. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 711–740. Springer, Heidelberg, November 2022.

AJ15. Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 308–326. Springer, Heidelberg, August 2015.

ALdP11. Nuttapong Attrapadung, Benoît Libert, and Elie de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 90–108. Springer, Heidelberg, March 2011.

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Heidelberg, August 2016.

AS17. Prabhanjan Ananth and Amit Sahai. Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part I*, volume 10210 of *LNCS*, pages 152–181. Springer, Heidelberg, April / May 2017.

BBL17. Fabrice Benhamouda, Florian Bourse, and Helger Lipmaa. CCA-secure inner-product functional encryption from projective hash functions. In Serge Fehr, editor, *PKC 2017, Part II*, volume 10175 of *LNCS*, pages 36–66. Springer, Heidelberg, March 2017.

BCFG17. Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer, Heidelberg, August 2001.

BIK+17. Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 1175–1191. ACM Press, October / November 2017.

BJK15. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Heidelberg, November / December 2015.

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

BV15. Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In Venkatesan Guruswami, editor, *56th FOCS*, pages 171–190. IEEE Computer Society Press, October 2015.

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

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

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

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

CLL+13. Jie Chen, Hoon Wei Lim, San Ling, Huaxiong Wang, and Hoeteck Wee. Shorter IBE and signatures via asymmetric pairings. In Michel Abdalla and Tanja Lange, editors, *PAIRING 2012*, volume 7708 of *LNCS*, pages 122–140. Springer, Heidelberg, May 2013.

CLT18. Guilhem Castagnos, Fabien Laguillaumie, and Ida Tucker. Practical fully secure unrestricted inner product functional encryption modulo p. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 733–764. Springer, Heidelberg, December 2018.

Coc01. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *LNCS*, pages 360–363. Springer, Heidelberg, December 2001.

DDM16. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 164–195. Springer, Heidelberg, March 2016.

DDM17. Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Strongly full-hiding inner product encryption. *Theoretical Computer Science*, 2017.

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

EHK+13. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013.

FHKP13. Eduarda S. V. Freire, Dennis Hofheinz, Eike Kiltz, and Kenneth G. Paterson. Non-interactive key exchange. In Kaoru Kurosawa and Goichiro Hanaoka, editors, *PKC 2013*, volume 7778 of *LNCS*, pages 254–271. Springer, Heidelberg, February / March 2013.

Gay20. Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 95–120. Springer, Heidelberg, May 2020.

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

GKL+13. S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. `https://eprint.iacr.org/2013/774`.

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

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

KKS17. Sungwook Kim, Jinsu Kim, and Jae Hong Seo. A new approach for practical function-private inner product encryption. Cryptology ePrint Archive, Report 2017/004, 2017. `https://eprint.iacr.org/2017/004`.

KLM+18. Sam Kim, Kevin Lewi, Avradip Mandal, Hart Montgomery, Arnab Roy, and David J. Wu. Function-hiding inner product encryption is practical. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 544–562. Springer, Heidelberg, September 2018.

Lin17. Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.

LT19. Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 520–551. Springer, Heidelberg, December 2019.

LV16. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

LW10. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010*, volume 5978 of *LNCS*, pages 455–479. Springer, Heidelberg, February 2010.

NPP22. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access-control. In *Asiacrypt '22*. Springer-Verlag, 2022. `https://ia.cr/2022/215`.

NR97. Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

OSW07. Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM CCS 2007*, pages 195–203. ACM Press, October 2007.

OT10. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

OT12a. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 591–608. Springer, Heidelberg, April 2012.

OT12b. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure unbounded inner-product and attribute-based encryption. In Xiaoyun Wang and Kazue Sako, editors, *ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 349–366. Springer, Heidelberg, December 2012.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *CRYPTO'84*, volume 196 of *LNCS*, pages 47–53. Springer, Heidelberg, August 1984.

SV23. Elaine Shi and Nikhil Vanjani. Multi-Client Inner Product Encryption: Function-Hiding Instantiations Without Random Oracles. In *International Conference on Practice and Theory of Public-Key Cryptography (PKC)*, 2023. `https://nikhilvanjani.github.io/publication/mcipe/`.

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

SW21. Elaine Shi and Ke Wu. Non-interactive anonymous router. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 489–520. Springer, Heidelberg, October 2021.

TAO16. Junichi Tomida, Masayuki Abe, and Tatsuaki Okamoto. Efficient functional encryption for inner-product values with full-hiding security. In Matt Bishop and Anderson C. A. Nascimento, editors, *ISC 2016*, volume 9866 of *LNCS*, pages 408–425. Springer, Heidelberg, September 2016.

Wat09. Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 619–636. Springer, Heidelberg, August 2009.

A Additional Definitions

A.1 Function-Hiding Decentralized Multi-Client FE

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

Definition 17 (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 $\mathsf{Param}_1 \times \cdots \times \mathsf{Param}_n$[5]. 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 [CDG+18a, CDG+18b, ABKW19, ABG19, LT19, CDSG+20]) 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 [CDG+18a]) 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.

[5] 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.

Security. We define the *function-hiding* security for DMCFE. In the seminal work by Chotard *et al.* [CDG+18a] and its follow-up study [CDSG+20], 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.* [AGT21b] abstracted out DMCFE into the notion of *Multi-Party Functional Encryption* (MPFE). The authors of [AGT21b] 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.

Definition 18 (Function-Hiding Security). *For a DMCFE scheme $\mathcal{E}$, a function class $\mathcal{F}$ and a* ppt *adversary $\mathcal{A}$ we define the experiment $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ as shown in Figure 3 and set $\mathcal{H} \coloneqq [n] \setminus \mathcal{C}$. 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. *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 $y_i^{(0)} \neq y_i^{(1)}$.*
2. *(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 *if for all* ppt *adversaries $\mathcal{A}$,*

$$\mathbf{Adv}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \coloneqq \left| \Pr\left[\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right] - \frac{1}{2} \right|$$

is negligible in λ.

$\underline{\mathsf{Initialize}(1^\lambda)}$:
$b \xleftarrow{\$} \{0,1\}$
$\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{DKGen}} \leftarrow \varnothing$
$(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda)$
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 $\beta \leftarrow (b' \stackrel{?}{=} b)$
Else, return 0

Fig. 3: Security game $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ for Definition 18

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

1. *Security against Static Corruption:* The experiment $\mathbf{Exp}^{\mathsf{stat\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that all queries to the oracle $\mathcal{O}\mathsf{Corrupt}$ must be submitted before $\mathsf{Initialize}$ is called.
2. *Security against Selective Challenges:* The experiment $\mathbf{Exp}^{\mathsf{sel\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that all queries to the oracle $\mathcal{O}\mathsf{Enc}$ must be submitted before $\mathsf{Initialize}$ is called.
3. *One-time Security:* The experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ 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 $y_i^{(0)} = y_i^{(1)}$.
4. *Security against Complete Challenges:* The experiment $\mathbf{Exp}^{\mathsf{pos\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that we add the following condition for $\mathsf{adm}(\mathcal{A}) = 0$:
   3. There exists $\mathsf{tag} \in \mathsf{Tag}$ so that a query $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, x_i^{(0)}, x_i^{(1)})$ has been asked for some but not all $i \in \mathcal{H}$, or there exists $\mathsf{tag\text{-}f} \in \mathsf{Tag}$ such that a query $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag\text{-}f}, y_i^{(0)}, y_i^{(1)})$ has been asked for some but not all $i \in \mathcal{H}$.

   In other words, we require for an adversary $\mathcal{A}$ to be *admissible* that, for any tag, either $\mathcal{A}$ makes no encryption (resp. key) query or makes at least one encryption (resp. key) query for each slot $i \in \mathcal{H}$.
5. *Weakly Function-Hiding:* We can weaken the function-hiding property by changing condition 2 for $\mathsf{adm}(\mathcal{A}) = 0$. More specifically, we replace it by the following condition *2'*:
   2'. (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

$$f^{(0)}(x_1^{(0)}, \ldots, x_n^{(0)}) = f^{(1)}(x_1^{(0)}, \ldots, x_n^{(0)}) = f^{(1)}(x_1^{(1)}, \ldots, x_n^{(1)}) \tag{7}$$

   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 $\mathbf{Exp}^{\mathsf{wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

A.2 Pseudorandom Functions (PRF)

Let $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{K}$ be sets representing domain, range and key space, respectively. We assume that they are implicitly indexed by the security parameter λ. Furthermore, let $\mathcal{R}$ be the set of all functions with domain $\mathcal{X}$ and range $\mathcal{Y}$. A family of functions $\{F_K\}_{K\in\mathcal{K}}$ that consists of efficiently computable functions $F_K \colon \mathcal{X} \to \mathcal{Y}$ is called *pseudorandom* (PRF) if for any ppt adversary $\mathcal{A}$, the following advantage is negligible in λ:

$$\mathbf{Adv}^{\mathsf{prf}}_{F_K,\mathcal{A}}(1^\lambda) := \left|\Pr[\mathcal{A}^{F_K(\cdot)} = 1] - \Pr[\mathcal{A}^{R(\cdot)} = 1]\right| ,$$

where $K \xleftarrow{\$} \mathcal{K}$ and $R \xleftarrow{\$} \mathcal{R}$.
It is well-known that PRFs can be constructed under DDH, *e.g.* the Naor-Reingold construction [NR97].

A.3 Non-Interactive Key Exchange (NIKE)

A NIKE scheme $\mathcal{N} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{SharedKey})$ for a key space $\mathcal{K}$ is a tuple of three efficient algorithms defined as follows:

$\mathsf{Setup}(1^\lambda)$**:** On input the security parameter 1^λ, the algorithm outputs the public parameters pp.
$\mathsf{KeyGen}(\mathsf{pp})$**:** On input the public parameters pp, the algorithm outputs a pair $(\mathsf{sk}, \mathsf{pk})$ consisting of a secret key sk and the corresponding public key pk.
$\mathsf{SharedKey}(\mathsf{sk}, \mathsf{pk}')$**:** On input a secret key sk and a (usually non-corresponding) public key pk', the algorithm deterministically outputs a shared key $K \in \mathcal{K}$.

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

$$\Pr\left[K_{i,j} = K_{j,i} \;\middle|\; \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mathsf{pk}_1, \mathsf{sk}_1), (\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathsf{KeyGen}(\mathsf{pp}), \\ K_{1,2} \leftarrow \mathsf{SharedKey}(\mathsf{sk}_1, \mathsf{pk}_2), \\ K_{2,1} \leftarrow \mathsf{SharedKey}(\mathsf{sk}_2, \mathsf{pk}_1) \end{array} \right] = 1 \ ,$$

where the probability is taken over the random coins of the algorithms.
Security. For a NIKE scheme $\mathcal{N}$ and a ppt adversary $\mathcal{A}$ we define the experiment $\mathbf{Exp}^{\mathsf{nike}\text{-}b}_{\mathcal{N},\mathcal{A}}$ as shown in Figure 4. The oracles $\mathcal{O}\mathsf{HonestGen}$, $\mathcal{O}\mathsf{Reveal}$, $\mathcal{O}\mathsf{Test}$ and $\mathcal{O}\mathsf{Corrupt}$ can be called in any order and any number of times. The adversary $\mathcal{A}$ is NOT admissible, denoted by $\mathsf{adm}(\mathcal{A}) = 0$, if either one of the following holds:

1. There exist public keys pk_1 and pk_2 such that $\mathcal{A}$ made the following queries
   - $\mathcal{O}\mathsf{Corrupt}(\mathsf{pk}_1)$,
   - $\mathcal{O}\mathsf{Test}(\mathsf{pk}_1, \mathsf{pk}_2)$ or $\mathcal{O}\mathsf{Test}(\mathsf{pk}_2, \mathsf{pk}_1)$'
2. There exist public keys pk_1 and pk_2 such that $\mathcal{A}$ made the following queries
   - $\mathcal{O}\mathsf{Reveal}(\mathsf{pk}_1, \mathsf{pk}_2)$ or $\mathcal{O}\mathsf{Reveal}(\mathsf{pk}_2, \mathsf{pk}_1)$,
   - $\mathcal{O}\mathsf{Test}(\mathsf{pk}_1, \mathsf{pk}_2)$ or $\mathcal{O}\mathsf{Test}(\mathsf{pk}_2, \mathsf{pk}_1)$.

Otherwise, we say that $\mathcal{A}$ is admissible and write $\mathsf{adm}(\mathcal{A}) = 1$. We call $\mathcal{N}$ *IND-secure* if for any ppt adversary $\mathcal{A}$, the following advantage is negligible in λ:

$$\mathbf{Adv}^{\mathsf{nike}}_{\mathcal{N},\mathcal{A}}(1^\lambda) := \left| \Pr\left[\mathbf{Exp}^{\mathsf{nike}\text{-}1}_{\mathcal{N},\mathcal{A}}(1^\lambda) = 1 \right] - \Pr\left[\mathbf{Exp}^{\mathsf{nike}\text{-}0}_{\mathcal{N},\mathcal{A}}(1^\lambda) = 1 \right] \right| \ .$$

NIKE can be constructed based on a variant of the Decisional Bilinear Diffie-Hellman assumption in the standard model [FHKP13, Section 4.3].

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

Definition 19. *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.*

It can be shown straightforwardly that given a cyclic group $\mathbb{G}$ and q, we have $\mathbf{Adv}^{\mathsf{DSDH}}_{\mathbb{G}}(1^\lambda) \leq 2 \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}}(1^\lambda)$.

$\underline{\mathsf{Initialize}(1^\lambda)}$:
$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$; $\mathcal{H} \leftarrow \varnothing$
Return pp

$\underline{\mathcal{O}\mathsf{HonestGen}()}$:
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}$; $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\mathsf{sk}, \mathsf{pk})\}$
Return pk

$\underline{\mathcal{O}\mathsf{Reveal}(\mathsf{pk}_1, \mathsf{pk}_2)}$:
If $\exists \mathsf{sk}_1$ s.t. $(\mathsf{sk}_1, \mathsf{pk}_1) \in \mathcal{H}$,
  return $K \leftarrow \mathsf{SharedKey}(\mathsf{sk}_1, \mathsf{pk}_2)$
If $\exists \mathsf{sk}_2$ s.t. $(\mathsf{sk}_2, \mathsf{pk}_2) \in \mathcal{H}$,
  return $K \leftarrow \mathsf{SharedKey}(\mathsf{sk}_2, \mathsf{pk}_1)$
Return $\perp$

$\underline{\mathcal{O}\mathsf{Test}(\mathsf{pk}_1, \mathsf{pk}_2)}$:
If $\{(\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2)\} \notin \mathcal{H}$,
  return $\perp$
If $b = 0$, return $K \xleftarrow{\$} \mathcal{K}$
Else, return $K \leftarrow \mathsf{SharedKey}(\mathsf{sk}_1, \mathsf{pk}_2)$

$\underline{\mathcal{O}\mathsf{Corrupt}(\mathsf{pk})}$:
Recover sk s.t. $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{H}$
$\mathcal{H} \leftarrow \mathcal{H} \setminus \{(\mathsf{sk}, \mathsf{pk})\}$
Return sk

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

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

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 $\mathbf{B} \coloneqq B \cdot \mathbf{U}$, which means:

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

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 are taken from the identity matrix I_N.

B Supporting Materials - Deferred Proofs

B.1 From One-Challenge to Multi-Challenge - Proof of Lemma 8

Lemma 8. *Let $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a DDFE scheme for the function class $\mathcal{F}$. If $\mathcal{E}$ is single-challenge weakly function-hiding, then it is also weakly function-hiding. More specifically, for any* ppt *adversary $\mathcal{A}$, there exists a* ppt *algorithm $\mathcal{B}$ such that*

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

where q_e and q_k denote the maximum numbers of different m_{pub} and k_{pub} that $\mathcal{A}$ can query to $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{KeyGen}$ respectively, and $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}, \mathsf{pos}\}$.

Proof. Let $\mathcal{A}$ be a ppt adversary in the experiment $\mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ and $b \xleftarrow{\$} \{0,1\}$ be the challenge bit. We denote the q_e distinct m_{pub} that can occur in a query to $\mathcal{O}\mathsf{Enc}$ by $m^{(1)}_{\text{pub}}, ..., m^{(q_e)}_{\text{pub}}$. Similarly, we denote the q_k distinct k_{pub} that can occur in queries to $\mathcal{O}\mathsf{KeyGen}$ by $k^{(1)}_{\text{pub}}, ..., k^{(q_k)}_{\text{pub}}$. We define a sequence of hybrid games:

Game $\mathsf{G}_{1,j}$ for $j \in [0; q_k]$: This hybrid is the same as $\mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ except that a query $\mathcal{O}\mathsf{KeyGen}(i, (k^0_{\text{pri}}, k^{(\ell)}_{\text{pub}}), (k^1_{\text{pri}}, k^{(\ell)}_{\text{pub}}))$ is answered by a partial decryption key for $(k^1_{\text{pri}}, k^{(\ell)}_{\text{pub}})$ (as opposed to $(k^0_{\text{pri}}, k^{(\ell)}_{\text{pub}})$) if $\ell \leq j$. Note that $\mathsf{G}_{1,0} = \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, conditioned on $b = 0$ as the challenge bit. The indistinguishability between $\mathsf{G}_{1,j}$ and $\mathsf{G}_{1,j-1}$ for $j \in [q_k]$ is proven in Lemma 20.

Game $\mathsf{G}_{2,j}$ for $j \in [0; q_e]$: This hybrid is the same as G_{1,q_k} except that a query $\mathcal{O}\mathsf{Enc}(i, (m^0_{\text{pri}}, m^{(\ell)}_{\text{pub}}), (m^1_{\text{pri}}, m^{(\ell)}_{\text{pub}}))$ is answered by an encryption of $(m^1_{\text{pri}}, m^{(\ell)}_{\text{pub}})$ (as opposed to $(m^0_{\text{pri}}, m^{(\ell)}_{\text{pub}})$) if $\ell \leq j$. Note that $\mathsf{G}_{2,0} = \mathsf{G}_{1,q_k}$ and $\mathsf{G}_{2,q_e} = \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, conditioned on $b = 1$ as the challenge bit. The indistinguishability between $\mathsf{G}_{2,j}$ and $\mathsf{G}_{2,j-1}$ for $j \in [q_e]$ is proven in Lemma 21.

For any hybrid $\mathsf{G}_{t,j}$, with $t \in [2], j \in [0, q_e] \cup [0, q_k]$, we define the event $\mathsf{G}_{t,j} = 1$ to indicate that $\mathcal{A}$ outputs 1 in $\mathsf{G}_{t,j}$. We calculate the advantage as follows:

$$
\begin{aligned}
&\mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \\
&= \frac{1}{2} \cdot \Big| \Pr\big[\mathcal{A} \text{ outputs 1 in } \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \mid b = 1\big] \\
&\qquad - \Pr\big[\mathcal{A} \text{ outputs 1 in } \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \mid b = 0\big] \Big| \\
&= \frac{1}{2} \cdot \big| \Pr\left[\mathsf{G}_{2,q_e} = 1\right] - \Pr\left[\mathsf{G}_{1,0} = 1\right] \big| \\
&= \frac{1}{2} \cdot \left| \sum_{j=1}^{q_k} \big(\Pr\left[\mathsf{G}_{1,j} = 1\right] - \Pr\left[\mathsf{G}_{1,j-1} = 1\right] \big) + \sum_{j=1}^{q_e} \big(\Pr\left[\mathsf{G}_{2,j} = 1\right] - \Pr\left[\mathsf{G}_{2,j-1} = 1\right] \big) \right| \\
&\leq \frac{1}{2} \cdot \left(\sum_{j=1}^{q_k} \big| \Pr\left[\mathsf{G}_{1,j} = 1\right] - \Pr\left[\mathsf{G}_{1,j-1} = 1\right] \big| + \sum_{j=1}^{q_e} \big| \Pr\left[\mathsf{G}_{2,j} = 1\right] - \Pr\left[\mathsf{G}_{2,j-1} = 1\right] \big| \right) \\
&\leq (q_k + q_e) \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \ ,
\end{aligned}
$$

where the last inequality is a consequence of Lemmas 20 and 21. □

Lemma 20. *If $\mathcal{E}$ is weakly function-hiding, then we have for each $j \in [q_k]$ that*

$$\big| \Pr\left[\mathsf{G}_{1,j} = 1\right] - \Pr\left[\mathsf{G}_{1,j-1} = 1\right] \big| \leq 2 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda) \ .$$

Proof. Let $\mathcal{A}$ be an adversary trying to distinguish between $\mathsf{G}_{1,j}$ and $\mathsf{G}_{1,j-1}$. We construct a ppt adversary $\mathcal{B}$ playing against $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ that uses black-box access to $\mathcal{A}$. $\mathcal{B}$ simulates the view of $\mathcal{A}$ as follows:

- *Initialization:* Upon $\mathcal{A}$ calling $\mathsf{Initialize}(1^\lambda)$, $\mathcal{B}$ runs the initialization procedure

$$\mathsf{Initialize}(1^\lambda, k^*_{\text{pub}} := k^{(j)}_{\text{pub}}, m^*_{\text{pub}} := m^{(j)}_{\text{pub}})$$

  of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.

- *Encryption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i,(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}),(m^1_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}))$, $\mathcal{B}$ queries the oracle $\mathcal{O}\mathsf{Enc}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ on input $(i,\mathsf{tag}_\ell,(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}),(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}))$ and forwards the response to $\mathcal{A}$. We emphasize that the returned ciphertext will be on the 0-side in this sequence of hybrids.
- *Key-generation Queries:*
  Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}$ on input $(i,(k^0_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}),(k^1_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}))$, $\mathcal{B}$ does:
  1. If $\ell<j$, $\mathcal{B}$ queries $(i,(k^1_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}),(k^1_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}))$ to the oracle $\mathcal{O}\mathsf{KeyGen}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.
  2. If $\ell=j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{KeyGen}(i,(k^0_{\mathrm{pri}},k^{(j)}_{\mathrm{pub}}),(k^1_{\mathrm{pri}},k^{(j)}_{\mathrm{pub}}))$ and forwards the response to $\mathcal{A}$.
  3. If $\ell>j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{KeyGen}(i,(k^0_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}),(k^0_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}))$ and forwards the response to $\mathcal{A}$.
- *Corruption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Corrupt}(i)$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Corrupt}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ on the same input i and forwards the response to $\mathcal{A}$.
- *Finalize:* Upon $\mathcal{A}$ 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 (more precisely, the first equality of 7), $\mathcal{A}$ is an admissible adversary in $\mathsf{G}_{1,j}$ and $\mathsf{G}_{1,j-1}$ if and only if $\mathcal{B}$ is an admissible adversary against $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$. Moreover, we observe that $\mathcal{B}$ simulates $\mathsf{G}_{1,j-1}$ to $\mathcal{A}$ if $b=0$, and $\mathsf{G}_{1,j}$ otherwise. Thus, we calculate

$$\begin{aligned}\left|\Pr\left[\mathsf{G}_{1,j}=1\right]-\Pr\left[\mathsf{G}_{1,j-1}=1\right]\right| &= \Big|\Pr\big[\mathcal{B}\text{ outputs 1 in }\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)\mid b=1\big]\\ &\quad-\Pr\big[\mathcal{B}\text{ outputs 1 in }\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)\mid b=0\big]\Big|\\ &\le 2\cdot\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)\end{aligned}$$

and the lemma is concluded. □

Lemma 21. *If $\mathcal{E}$ is weakly function-hiding, then we have for each $j\in[q_e]$ that*

$$\left|\Pr\left[\mathsf{G}_{2,j}=1\right]-\Pr\left[\mathsf{G}_{2,j-1}=1\right]\right|\le 2\cdot\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)\ .$$

Proof. Let $\mathcal{A}$ be an adversary trying to distinguish between $\mathsf{G}_{2,j}$ and $\mathsf{G}_{2,j-1}$. We construct a ppt adversary $\mathcal{B}$ playing against $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ that uses black-box access to $\mathcal{A}$. $\mathcal{B}$ simulates the view of $\mathcal{A}$ as follows:

- *Initialization:* Upon $\mathcal{A}$ calling $\mathsf{Initialize}(1^\lambda)$, $\mathcal{B}$ runs the initialization procedure
  $$\mathsf{Initialize}(1^\lambda,k^*_{\mathrm{pub}}:=k^{(j)}_{\mathrm{pub}},m^*_{\mathrm{pub}}:=m^{(j)}_{\mathrm{pub}})$$
  of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.
- *Encryption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i,(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}),(m^1_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}))$, $\mathcal{B}$ behaves as follows:
  1. If $\ell<j$, $\mathcal{B}$ queries $(i,(m^1_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}),(m^1_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}))$ to the oracle $\mathcal{O}\mathsf{Enc}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.
  2. If $\ell=j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Enc}(i,(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}),(m^1_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}))$ and forwards the response to $\mathcal{A}$.
  3. If $\ell>j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Enc}(i,(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}),(m^0_{\mathrm{pri}},m^{(\ell)}_{\mathrm{pub}}))$ and forwards the response to $\mathcal{A}$.
- *Key-generation Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}(i,(k^0_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}),(k^1_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}))$, $\mathcal{B}$ queries $(i,(k^1_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}),(k^1_{\mathrm{pri}},k^{(\ell)}_{\mathrm{pub}}))$ the oracle $\mathcal{O}\mathsf{Enc}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$. We emphasize that the returned key components will be on the 1-side in this sequence of hybrids.
- *Corruption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Corrupt}(i)$ for some $i\in[n]$, $\mathcal{B}$ queries the oracle $\mathcal{O}\mathsf{Corrupt}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$ on the same input i and forwards the response $(\mathsf{ek}_i,\mathsf{sk}_i)$ to $\mathcal{A}$.

- *Finalize:* Upon $\mathcal{A}$ 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 (more precisely, the second equality of 7), $\mathcal{A}$ is an admissible adversary in $\mathsf{G}_{2,j}$ and $\mathsf{G}_{2,j-1}$ if and only if $\mathcal{B}$ is an admissible adversary against $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$. Moreover, we observe that $\mathcal{B}$ simulates $\mathsf{G}_{2,j-1}$ if $b=0$, and $\mathsf{G}_{2,j}$ otherwise. Using the same calculation as in Lemma 20, we conclude that

$$\left|\Pr\left[\mathsf{G}_{2,j}=1\right]-\Pr\left[\mathsf{G}_{2,j-1}=1\right]\right| \leq 2\cdot \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{B}}(1^\lambda)$$

and the proof is completed. □

B.2 From Weakly to Fully Function-Hiding - Proof of Lemma 9

Lemma 9. *If there exists a weakly function-hiding DDFE scheme $\mathcal{E}$ for $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$, then there exists a (fully) function-hiding DDFE scheme $\mathcal{E}'$ for $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$. More precisely, for any* ppt *adversary $\mathcal{A}$, there exists a* ppt *algorithm $\mathcal{B}$ such that*

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

where $\mathsf{xxx}\subseteq\{\mathsf{stat},\mathsf{sel},\mathsf{1chal},\mathsf{pos}\}$.

Proof. Given $\mathcal{E}=(\mathsf{GSetup},\mathsf{LSetup},\mathsf{KeyGen},\mathsf{Enc},\mathsf{Dec})$, we define the fully function-hiding scheme $\mathcal{E}'=(\mathsf{GSetup}',\mathsf{LSetup}',\mathsf{KeyGen}',\mathsf{Enc}',\mathsf{Dec}')$ for $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ as follows:

- *GSetup:* $\mathsf{GSetup}'(1^\lambda)$ runs $(\mathsf{pp})\leftarrow\mathsf{GSetup}(1^\lambda)$ and outputs the public parameters pp.
- *LSetup:* $\mathsf{LSetup}'(\mathsf{pp})$ runs $(\mathsf{pk}_i,\mathsf{sk}_i)\leftarrow\mathsf{LSetup}(\mathsf{pp})$ and outputs $(\mathsf{pk}'_i,\mathsf{sk}'_i):=(\mathsf{pk}_i,\mathsf{sk}_i)$.
- *Key Generation:* $\mathsf{KeyGen}'(\mathsf{sk}'_i,k_i=(\mathbf{y}_i,\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i))$ parses $\mathsf{sk}'_i=\mathsf{sk}_i$, runs $\mathsf{dk}_i\leftarrow\mathsf{KeyGen}(\mathsf{sk}_i,k=(\mathbf{y}_i\parallel 0^{N_i},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i))$ and outputs $\mathsf{dk}'_i:=\mathsf{dk}_i$.
- *Encryption:* $\mathsf{Enc}'(\mathsf{ek}'_i,m_i=(\mathbf{x}_i,\mathcal{U}_{M,i},\mathsf{tag}_i))$ parses $\mathsf{ek}'_i=\mathsf{ek}_i$, computes a ciphertext $\mathsf{ct}_i\leftarrow\mathsf{Enc}(\mathsf{ek}_i,m_i=(\mathbf{x}_i\parallel 0^{N_i},\mathcal{U}_{M,i},\mathsf{tag}_i))$ and outputs $\mathsf{ct}'_i:=\mathsf{ct}_i$.
- *Decryption:* $\mathsf{Dec}((\mathsf{dk}'_i)_{i\in\mathcal{U}_K},(\mathsf{ct}'_i)_{i\in\mathcal{U}_M})$ outputs $d\leftarrow\mathsf{Dec}((\mathsf{dk}'_i)_{i\in\mathcal{U}_K},(\mathsf{ct}'_i)_{i\in\mathcal{U}_M})$.

The correctness of $\mathcal{E}'$ follows immediately from that of $\mathcal{E}$ and the fact that

$$\left\langle\left(\mathbf{x}_i\parallel 0^{N_i}\right)_{i\in[n]},\left(\mathbf{y}_i\parallel 0^{N_i}\right)_{i\in[n]}\right\rangle=\left\langle(\mathbf{x}_i)_{i\in[n]},(\mathbf{y}_i)_{i\in[n]}\right\rangle \ ,$$

where we denote $(\mathbf{z}_i)_{i\in[n]}:=(\mathbf{z}_1\parallel\ldots\parallel\mathbf{z}_n)$ for arbitrary vectors $\mathbf{z}_1,\ldots,\mathbf{z}_n$. Furthermore, we show that $\mathcal{E}'$ enjoys the function-hiding property. Towards this, we consider a sequence of hybrid games $\mathsf{G}_0,\ldots,\mathsf{G}_3$ where G_0 equals $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}',\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda)$, where the challenge bit is 0, and G_3 equals $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}',\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda)$, where the challenge bit is 1, and $\mathcal{A}$ is a ppt adversary. For $i\in[3]$, we denote the event $\mathsf{G}_i=1$ to signify that $\mathcal{A}$ outputs 1 in the hybrid G_i.

Game G_0: This is $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}',\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda)$ conditioned on the challenge bit $b=0$. We recall that in this specific functionality $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$, there is the concept of tags in the public information of keys and ciphertexts. 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 $\mathcal{O}\mathsf{KeyGen}$. Queries to $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{KeyGen}$ are answered as follows:

- Upon $\mathcal{A}$ querying
$$\mathcal{O}\mathsf{Enc}(i, (\mathbf{x}_i^{(0)}, \mathcal{U}_{M,i}, \mathsf{tag}_i), (\mathbf{x}_i^{(1)}, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
the challenger queries to its weakly function-hiding oracle for
$$\mathsf{ct}_{\ell,i} \leftarrow \mathsf{Enc}(i, (\mathbf{x}_i^{(0)} \parallel 0^{N_i}, \mathcal{U}_{M,i}, \mathsf{tag}_i), (\mathbf{x}_i^{(0)} \parallel 0^{N_i}, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
and returns $\mathsf{ct}'_{\ell,i} \coloneqq \mathsf{ct}_{\ell,i}$.
- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}(i, (\mathbf{y}_i^{(0)}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i), (\mathbf{y}_i^{(1)}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))$, the challenger queries to its weakly function-hiding oracle for
$$\mathsf{dk}_{k,i} \leftarrow \mathsf{KeyGen}(i, (\mathbf{y}_i^{(0)} \parallel 0^{N_i}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i), (\mathbf{y}_i^{(0)} \parallel 0^{N_i}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))$$
and returns $\mathsf{dk}'_{k,i} \coloneqq \mathsf{dk}_{k,i}$.

We note that N_i are included in pk_i which can be known to the simulator via queries to $\mathcal{O}\mathsf{HonestGen}$, upon requests from $\mathcal{A}$. In other words, the ciphertexts $(\mathsf{ct}'_{\ell,i})_{i\in\mathcal{U}_M}$ encrypt the vector $(\mathbf{x}_i^{(0)} \parallel 0^{N_i})_{i\in\mathcal{U}_M}$, and the partial decryption keys $(\mathsf{dk}'_{k,i})_{i\in\mathcal{U}_K}$ allow for the computation of the inner product with the vector $(\mathbf{y}_i^{(0)} \parallel 0^{N_i})_{i\in\mathcal{U}_K}$.

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

- Upon $\mathcal{A}$ querying
$$\mathcal{O}\mathsf{Enc}(i, (\mathbf{x}_i^{(0)}, \mathcal{U}_{M,i}, \mathsf{tag}_i), (\mathbf{x}_i^{(1)}, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
the challenger queries to its weakly function-hiding oracle for
$$\mathsf{ct}_{\ell,i} \leftarrow \mathsf{Enc}(i, (0^{N_i} \parallel \mathbf{x}_i^{(1)}, \mathcal{U}_{M,i}, \mathsf{tag}_i), (0^{N_i} \parallel \mathbf{x}_i^{(1)}, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
and returns $\mathsf{ct}'_{\ell,i} \coloneqq \mathsf{ct}_{\ell,i}$.
- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}(i, (\mathbf{y}_i^{(0)}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i), (\mathbf{y}_i^{(1)}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))$, the challenger queries to its weakly function-hiding oracle for
$$\mathsf{dk}_{k,i} \leftarrow \mathsf{KeyGen}(i, (\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i), (\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))$$
and returns $\mathsf{dk}'_{k,i} \coloneqq \mathsf{dk}_{k,i}$.

Thus, the ciphertexts $(\mathsf{ct}'_{\ell,i})_{i\in\mathcal{U}_M}$ encrypt the vector $(0^{N_i} \parallel \mathbf{x}_i^{(1)})_{i\in\mathcal{U}_M}$ (as opposed to $(\mathbf{x}_i^{(0)} \parallel 0^{N_i})_{i\in\mathcal{U}_M}$ in G_0), and the partial decryption keys $(\mathsf{dk}'_{k,i})_{i\in\mathcal{U}_K}$ allow for the computation of the inner product with the vector $(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$ (as opposed to $(\mathbf{y}_i^{(0)} \parallel 0^{N_i})_{i\in\mathcal{U}_K}$ in G_0). Let $n \coloneqq |\mathcal{U}_K| = |\mathcal{U}_M|$ in case of correct evaluation in $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$. The admissibility of $\mathcal{A}$ states that $\langle (\mathbf{x}_i^{(0)})_{i\in[n]}, (\mathbf{y}_i^{(0)})_{i\in[n]} \rangle = \langle (\mathbf{x}_i^{(1)})_{i\in[n]}, (\mathbf{y}_i^{(1)})_{i\in[n]} \rangle$ which implies that

$$\begin{aligned}\left\langle \left(\mathbf{x}_i^{(0)} \parallel 0^{N_i}\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel 0^{N_i}\right)_{i\in[n]} \right\rangle &= \left\langle \left(\mathbf{x}_i^{(0)} \parallel 0^{N_i}\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle \\ &= \left\langle \left(0^{N_i} \parallel \mathbf{x}_i^{(1)}\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle\end{aligned}$$

And our simulator's queries are admissible in the weakly function-hiding model. Then it follows by the weak function-hiding property of $\mathcal{E}'$ that there exists a ppt adversary $\mathcal{B}$ such that

$$\begin{aligned}|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1]| = \Big| &\Pr\left[\mathcal{B} \text{ outputs 1 in } \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{B}}(1^\lambda) \mid b = 1\right] \\ &- \Pr\left[\mathcal{B} \text{ outputs 1 in } \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{B}}(1^\lambda) \mid b = 0\right]\Big| \\ &\leq 2 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{B}}(1^\lambda)\end{aligned}$$

The simulator's queries change only the function's contents while relaying $\mathcal{U}_K, \mathsf{tag\text{-}f}_k$ queried by $\mathcal{A}$. The same will hold for the following hybrids.

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

- Upon $\mathcal{A}$ querying
$$\mathcal{O}\mathsf{Enc}(i,(\mathbf{x}_i^{(0)},\mathcal{U}_{M,i},\mathsf{tag}_i),(\mathbf{x}_i^{(1)},\mathcal{U}_{M,i},\mathsf{tag}_i))$$
the challenger queries to its weakly function-hiding oracle for
$$\mathsf{ct}_{\ell,i}\leftarrow\mathsf{Enc}(i,(\mathbf{x}_i^{(1)}\parallel 0^{N_i},\mathcal{U}_{M,i},\mathsf{tag}_i),(\mathbf{x}_i^{(1)}\parallel 0^{N_i},\mathcal{U}_{M,i},\mathsf{tag}_i))$$
and returns $\mathsf{ct}'_{\ell,i}\coloneqq\mathsf{ct}_{\ell,i}$.
- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}(i,(\mathbf{y}_i^{(0)},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i),(\mathbf{y}_i^{(1)},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i))$, the challenger queries to its weakly function-hiding oracle for
$$\mathsf{dk}_{k,i}\leftarrow\mathsf{KeyGen}(i,(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i),(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i))$$
and returns $\mathsf{dk}'_{k,i}\coloneqq\mathsf{dk}_{k,i}$.

That is, the challenger provides a ciphertext of $(\mathbf{x}_i^{(1)}\parallel 0^{N_i})_{i\in\mathcal{U}_M}$ and a decryption key for $(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$, as opposed to $(0^{N_i}\parallel\mathbf{x}_i^{(1)})_{i\in\mathcal{U}_M}$ and $(\mathbf{y}_i^{(0)}\parallel\mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$ in G_1. Let $n\coloneqq|\mathcal{U}_K|=|\mathcal{U}_M|$ in case of correct evaluation in $\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}}$. Notice that

$$\begin{aligned}\left\langle\left(0^{N_i}\parallel\mathbf{x}_i^{(1)}\right)_{i\in[n]},\left(\mathbf{y}_i^{(0)}\parallel\mathbf{y}_i^{(1)}\right)_{i\in[n]}\right\rangle&=\left\langle\left(0^{N_i}\parallel\mathbf{x}_i^{(1)}\right)_{i\in[n]},\left(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)}\right)_{i\in[n]}\right\rangle\\&=\left\langle\left(\mathbf{x}_i^{(1)}\parallel 0^{N_i}\right)_{i\in[n]},\left(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)}\right)_{i\in[n]}\right\rangle\ .\end{aligned}$$

And our simulator's queries are admissible in the weakly function-hiding model. Then it follows by the weak function-hiding property of $\mathcal{E}'$ that there exists a ppt adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_2=1]-\Pr[\mathsf{G}_1=1]|\le 2\cdot\mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}},\mathcal{B}}(1^\lambda)$.

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

- Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}(i,(\mathbf{y}_i^{(0)},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i),(\mathbf{y}_i^{(1)},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i))$, the challenger queries to its weakly function-hiding oracle for
$$\mathsf{dk}_{k,i}\leftarrow\mathsf{KeyGen}(i,(\mathbf{y}_i^{(1)}\parallel 0^{N_i},\mathcal{U}_{K,i},\mathsf{tag\text{-}f}_i),(\mathbf{y}_i^{(1)}\parallel 0^{N_i},\mathsf{tag\text{-}f}_i))$$
and returns $\mathsf{dk}'_{k,i}\coloneqq\mathsf{dk}_{k,i}$.

Thus, the challenger provides a decryption key for $(\mathbf{y}_i^{(1)}\parallel 0^{N_i})_{i\in\mathcal{U}_K}$, as opposed to $(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$ in G_2. Let $n\coloneqq|\mathcal{U}_K|=|\mathcal{U}_M|$ in case of correct evaluation in $\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}}$. We have

$$\left\langle\left(\mathbf{x}_i^{(1)}\parallel 0^{N_i}\right)_{i\in[n]},\left(\mathbf{y}_i^{(1)}\parallel\mathbf{y}_i^{(1)}\right)_{i\in[n]}\right\rangle=\left\langle\left(\mathbf{x}_i^{(1)}\parallel 0^{N_i}\right)_{i\in[n]},\left(\mathbf{y}_i^{(1)}\parallel 0^{N_i}\right)_{i\in[n]}\right\rangle\ .$$

And our simulator's queries are admissible in the weakly function-hiding model. As above, it follows by the weak function-hiding property of $\mathcal{E}'$ that there exists a ppt adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_3=1]-\Pr[\mathsf{G}_2=1]|\le 2\cdot\mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}},\mathcal{B}}(1^\lambda)$. Note that G_3 equals the experiment $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}',\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}},\mathcal{A}}(1^\lambda)$ conditioned on $b=1$.

Using a hybrid argument, we conclude that:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}',\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}},\mathcal{A}}(1^\lambda)=\frac{1}{2}|\Pr[\mathsf{G}_3=1]-\Pr[\mathsf{G}_0=1]|&\le\frac{1}{2}\cdot\sum_{i=1}^{3}|\Pr[\mathsf{G}_i=1]-\Pr[\mathsf{G}_{i-1}=1]|\\&\le 3\cdot\mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}_{\mathsf{dyn}}^{\mathsf{ip}},\mathcal{B}}(1^\lambda)\end{aligned}$$

and the lemma is proved. □

B.3 Proof of Lemma 11 - Single Secret-Sharing Swapping

Lemma 11 (Single Secret-Sharing Swapping). *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda), J = J(\lambda), N_i = N_i(\lambda) \in \mathbb{N}$ where $i \in H$ and $H, K, L, J, N_i : \mathbb{N} \to \mathbb{N}$ are polynomials. Let $(\mathbf{B}_i, \mathbf{B}_i^*)_{i\in[H]}$ be two random dual bases of dimension $4N_i + 4$ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$. All basis vectors are kept secret.*

For each $i \in [H]$, consider public vectors $(\mathbf{x}_i, \mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}_{\ell,i}^{(\mathsf{rep})'}, \mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)}, \mathbf{y}_{k,i}^{(\mathsf{rep})})_{k\in[K],\ell\in[L]}^{\mathsf{rep},j\in[J]}$ of length N_i such that $\sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(0,j)}\rangle = \sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(1,j)}\rangle$ and $\langle \mathbf{x}_i, \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}\rangle$ is a constant for all $j \in [J], i \in [H]$. Let $(R, R_k)_{j\in[J],k\in[K]}$ be some public scalars. Given $r, r_\ell, \rho_i, \rho_{\ell,i}^{(\mathsf{rep})}, \pi_i^{(j)}, \pi_{k,i}^{(\mathsf{rep})}, \sigma_i, \sigma_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ 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:*

$$D_0 := \left\{ \begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ \quad (\mathbf{u}_i = (\boxed{\mathbf{x}_i},\ \boxed{0^{N_i}},\ r,\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H]} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\};$$

$$D_1 := \left\{ \begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ \quad (\mathbf{u}_i = (\boxed{0^{N_i}},\ \boxed{\mathbf{x}_i},\ r,\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H]} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\}.$$

More specifically, we have:

$$\left| \Pr_{\mathsf{samp}\sim D_0}[\mathcal{A}(\mathsf{samp}) \to 1] - \Pr_{\mathsf{samp}\sim D_1}[\mathcal{A}(\mathsf{samp}) \to 1] \right| \leq (2N_i + 8) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

for any ppt $\mathcal{A}$ *with fixed* $\left(\mathbf{x}_i, (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}_{\ell,i}^{(\mathsf{rep})'})_{\ell=1}^L, R, \mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)}, (\mathbf{y}_{k,i}^{(\mathsf{rep})}, R_k)_{k=1}^K\right)_{i\in[H]}^{j\in[J]}$.

Proof (of Lemma 11). 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 $2N_i + 1$ of $(\mathbf{u}_i)_{i=1}^H$ and of $\sigma_i \xleftarrow{\$} \mathbb{Z}_q$ at coordinate $2N_i + 1$ of $(\mathbf{v}_i)_{i=1}^H$ so as to introduce a new non-zero $r' \xleftarrow{\$} \mathbb{Z}_q^*$ at coordinate $4N_i + 4$ in $(\mathbf{u}_i)_{i=1}^H$ and secret sharings $(\tau_i)_{i=1}^H$ of 0 with only non-zero τ_i at coordinate $4N_i + 4$ of $(\mathbf{v}_i^{(j)})_{i=1}^H$, where $j \in [J]$. We recall that for each $j \in [J]$, 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 $4N_i + 4$ of $\mathbf{u}_i$, while keeping $\mathbf{v}_i^{(j)}[4N_i + 4] = \mathbf{u}_{\ell,i}^{(\mathsf{rep})}[4N_i + 4] = \mathbf{v}_{k,i}^{(\mathsf{rep})}[4N_i + 4] = 0$. Given a DSDH instance $([\![a]\!]_1, [\![b]\!]_1, [\![c]\!]_1)$ in $\mathbb{G}_1$ where $\delta := c - ab$ is either 0 or 1, the basis changing matrices are:

$$\mathbf{B}_i = \begin{bmatrix} 1 & a \\ 0 & 1 \end{bmatrix}_{2N_i+1,4N_i+4} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}_{2N_i+1,4N_i+4} \cdot \mathbf{H}_i^* \ .$$

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

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

We cannot compute $\mathbf{b}^*_{i,2N_i+1}$ but can write the $\mathbf{v}$-vectors in $\mathbf{H}^*$ and observe that they stay invariant in $\mathbf{B}^*_i$ as the $(4N_i+4)$-th coordinate is 0:

$$\begin{aligned}\mathbf{v}^{(j)}_i &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{H}^*_i}\\ &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}^*_i}\\ \mathbf{v}^{(\mathsf{rep})}_{k,i} &= (\mathbf{y}^{(\mathsf{rep})}_{k,i},\ \mathbf{y}^{(\mathsf{rep})}_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{H}^*_i}\\ &= (\mathbf{y}^{(\mathsf{rep})}_{k,i},\ \mathbf{y}^{(\mathsf{rep})}_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}^*_i}\ .\end{aligned}$$

If $\delta = 0$ we are in G_0 else we are in $\mathsf{G}_{0.1}$, while updating r to $r + br'$[6]. The difference in advantages is $|\Pr[\mathsf{G}_{0.1}=1] - \Pr[\mathsf{G}_0=1]| \le 2\cdot\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

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, for every $j \in [J]$. Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta := c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}^*_i)$ are changed following:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0\\ -a & 1\end{bmatrix}_{2N_i+1,4N_i+4}\cdot\mathbf{H}_i;\quad \mathbf{B}^*_i = \begin{bmatrix} 1 & a\\ 0 & 1\end{bmatrix}_{2N_i+1,4N_i+4}\cdot\mathbf{H}^*_i\ .$$

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

$$\begin{aligned}\mathbf{v}^{(j)}_i &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}^*_i} + (0^{N_i},\ 0^{N_i},\ b\tau_i,\ 0,\ 0,\ 0^{N_i},\ 0^{N_i},\ c\tau_i)_{\mathbf{H}^*_i}\\ &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\boxed{\sigma_i+b\tau_i},\ \pi^{(j)}_i,\ 0,\ 0^{N_i},\ 0^{N_i},\boxed{\delta\tau_i})_{\mathbf{B}^*_i}\\ \mathbf{v}^{(\mathsf{rep})}_{k,i} &= (\mathbf{y}^{(\mathsf{rep})}_{k,i},\ \mathbf{y}^{(\mathsf{rep})}_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\ .\end{aligned}$$

For each $j \in [J]$, the secret shares $(\sigma_i)_{i=1}^H$ are updated to $(\sigma_i + b\tau_i)_{i=1}^H$ and still satisfy:

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

because $(\tau_i)_{i=1}^H$ is a secret sharing of 0. We cannot compute $\mathbf{b}_{i,4N_i+4}$ but can write the $\mathbf{u}$-vectors in $\mathbf{H}_i$, for $r'', r_\ell \xleftarrow{\$} \mathbb{Z}_q$, $r' \xleftarrow{\$} \mathbb{Z}^*_q$:

$$\begin{aligned}\mathbf{u}_i &= (\mathbf{x}_i,\ 0^{N_i},\ r'',\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ r')_{\mathbf{H}_i} = (\mathbf{x}_i,\ 0^{N_i},\ r''+ar',\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ r')_{\mathbf{B}_i}\\ \mathbf{u}^{(\mathsf{rep})}_{\ell,i} &= (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{H}_i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\ ,\end{aligned}$$

[6] It is thanks to the randomness of $r \xleftarrow{\$} \mathbb{Z}_q$ that allows us to update br' without changing the distribution. When applying this secret-sharing swapping lemma for our $\mathsf{FH\text{-}DMCFE}$ scheme, this random r is provided by the RO while hashing the tags.

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:

$$\mathbf{u}^{(\mathsf{rep})}_{\ell,i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{2N_i},\ 0)_{\mathbf{B}_i}; \qquad \mathbf{u}_i = (\mathbf{x}_i,\ 0,\ \boxed{r},\ 0,\ \rho_i,\ 0^{2N_i},\ \boxed{r'})_{\mathbf{B}_i}$$
$$\mathbf{v}^{(j)}_i = (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \boxed{\sigma_i},\ \pi^{(j)}_i,\ 0,\ 0^{2N_i},\ \boxed{\tau_i})_{\mathbf{B}^*_i}; \quad \mathbf{v}^{(\mathsf{rep})}_{k,i} = (\mathbf{y}^{(\mathsf{rep})}_{k,i},\ \mathbf{y}^{(\mathsf{rep})}_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0^{2N_i}, 0)_{\mathbf{B}^*_i}$$

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 $(\mathbf{y}^{(1,j)}_i, \mathbf{y}^{(0,j)}_i)$ (respectively $(\mathbf{y}^{(\mathsf{rep})}_{k,i}, \mathbf{y}^{(\mathsf{rep})}_{k,i})$) from coordinates $[1, N_i], [N_i + 1, 2N_i]$ to $[2N_i + 4, 3N_i + 3], [3N_i + 4, 4N_i + 3]$ of $\mathbf{v}^{(j)}_i$ (respectively of $\mathbf{v}^{(\mathsf{rep})}_{k,i}$). The bases are changed following using the following matrices (we denote $B_i[row, col]$ the entry at row row and column col of B_i)

$$B_i = \begin{cases} B_i[row, col] & = 1 \text{ if } row = col \\ B_i[row, col] & = 1 \text{ if } (row, col) \in \{(2N_i + 4 + d, 1 + d) : d \in [0, N_i - 1]\} \\ B_i[row, col] & = 1 \text{ if } (row, col) \in \{(3N_i + 4 + d, N_i + 1 + d) : d \in [0, N_i - 1]\} \\ B_i[row, col] & = 0 \text{ otherwise} \end{cases}$$
$$B'_i \coloneqq \left(B_i^{-1}\right)^\top$$
$$\mathbf{B}_i = B_i \cdot \mathbf{H}_i; \quad \mathbf{B}^*_i = B'_i \cdot \mathbf{H}^*_i \ .$$

We write the vectors as follows, observing that the $\mathbf{u}$-vectors stay invariant because their coordinates $[2N_i + 4, 3N_i + 3], [3N_i + 4, 4N_i + 3]$ are all 0 and the duplication is done correctly for the $\mathbf{v}$-vectors:

$$\mathbf{u}^{(\mathsf{rep})}_{\ell,i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{H}_i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}$$
$$\mathbf{u}_i = (\mathbf{x}_i,\ 0,\ r,\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ r')_{\mathbf{H}_i} = (\mathbf{x}_i,\ 0,\ r,\ 0,\ \rho_i,\ 0^{N_i},\ 0^{N_i},\ r')_{\mathbf{B}_i}$$
$$\mathbf{v}^{(j)}_i = (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ 0^{N_i},\ 0^{N_i},\ \tau_i)_{\mathbf{H}^*_i} = (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ \boxed{\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i},\ \tau_i)_{\mathbf{B}^*_i}$$
$$\mathbf{v}^{(\mathsf{rep})}_{k,i} = (\mathbf{y}^{(\mathsf{rep})}_{k,i},\ \mathbf{y}^{(\mathsf{rep})}_{k,i},\ \sigma_{k,i},\ \pi_{k,i},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{H}^*_i} = (\mathbf{y}^{(\mathsf{rep})}_{k,i}, \mathbf{y}^{(\mathsf{rep})}_{k,i}, \sigma_{k,i}, \pi_{k,i},\ 0, \boxed{\mathbf{y}^{(\mathsf{rep})}_{k,i}, \mathbf{y}^{(\mathsf{rep})}_{k,i}}, 0)_{\mathbf{B}^*_i} \ .$$

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 $\mathbf{x}_i$ from coordinate $[1, N_i]$ to $[2N_i{+}4, 3N_i{+}3]$ of $\mathbf{u}_i$, using the randomness ρ_i at coordinate $2N_i + 3$. We proceed by a sequence of $N_i + 1$ hybrids, namely $\mathsf{G}_{2.k}$ for $k \in [0, N_i]$, such that $\mathsf{G}_{2.0} = \mathsf{G}_2$ and in $\mathsf{G}_{2.k}$ the first coordinates $[1, k]$ are swapped to $[2N_i + 4, 2N_i + 3 + k]$, for $k \geq 1$. For $k \in [N_i]$, the transition from $\mathsf{G}_{2.k-1}$ to $\mathsf{G}_{2.k}$ is described below. 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:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 & 0 \\ -a & 1 & a \\ 0 & 0 & 1 \end{bmatrix}_{k, 2N_i+3, 2N_i+4+k-1} \cdot \mathbf{H}_i; \quad \mathbf{B}^*_i = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & -a & 1 \end{bmatrix}_{k, 2N_i+3, 2N_i+4+k-1} \cdot \mathbf{H}^*_i \ .$$

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

$$\begin{aligned}\mathbf{u}_i &= (\underbrace{0,..,0,\mathbf{x}_i[k],..,\mathbf{x}_i[N_i]}_{\text{first } (k-1)\text{-th coords are } 0},\ 0^{N_i},\ r,\ 0,\ \rho_i,\ \underbrace{\mathbf{x}_i[1],..,\mathbf{x}_i[k-1],0,..,0}_{\text{last } (N_i-k+1)\text{-th coords are } 0},\ 0^{N_i},\ r')_{\mathbf{B}_i}\\ &+ (\underbrace{0,..,0,-c\mathbf{x}_i[k],0,..,0}_{k\text{-th coord among } N_i},\ 0^{N_i},\ 0,\ 0,\ b\mathbf{x}_i[k],\ \underbrace{0,..,0,c\mathbf{x}_i[k],0,..,0}_{k\text{-th coord among } N_i},\ 0^{N_i},\ 0)_{\mathbf{H}_i}\\ &= (\underbrace{0,..,0,\boxed{\mathbf{x}_i[k]-\delta\mathbf{x}_i[k]},..,\mathbf{x}_i[N_i]}_{\text{first } (k-1)\text{-th coords are } 0},\ 0^{N_i},\ r,\ 0,\ \boxed{\rho_i+b\mathbf{x}_i[k]},\underbrace{\mathbf{x}_i[1],..,\mathbf{x}_i[k-1],\boxed{\delta\mathbf{x}_i},..,0}_{\text{last } (N_i-k)\text{-th coords are } 0},\ 0^{N_i},\ r')_{\mathbf{B}_i}\end{aligned}$$

$$\mathbf{u}^{(\mathsf{rep})}_{\ell,i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\ .$$

We cannot compute $\mathbf{b}^*_{i,1+k}$ and $\mathbf{b}^*_{1,2N_i+4+k-1}$ due to the lack of $[\![a]\!]_2$, but the $\mathbf{v}$-vectors can be written in $\mathbf{H}^*_i$ indeed they stay invariant: for instance we consider $\mathbf{v}_i$, the same holds for $\mathbf{v}^{(\mathsf{rep})}_{k,i}$

$$\begin{aligned}\mathbf{v}^{(j)}_i &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ \mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \tau_i)_{\mathbf{H}^*_i}\\ &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i, -a\mathbf{y}^{(1,j)}_i[k]+a\mathbf{y}^{(1,j)}_i[k],\ \mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \tau_i)_{\mathbf{B}^*_i}\\ &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ \mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \tau_i)_{\mathbf{B}^*_i}\ .\end{aligned}$$

If $\delta = 0$ we are in $\mathsf{G}_{2.k-1}$, else we are in $\mathsf{G}_{2.k}$, while updating ρ_i to $\rho_i + b\mathbf{x}_i[k]$ that stays uniformly random in $\mathbb{Z}_q$. We have $|\Pr[\mathsf{G}_{2.k}=1]-\Pr[\mathsf{G}_{2.k-1}=1]| \le 2\cdot\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$ and in the end $|\Pr[\mathsf{G}_3=1]-\Pr[\mathsf{G}_2=1]| \le 2N_i\cdot\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

The vectors, when we arrive at G_3, are of the form:

$$\begin{aligned}\mathbf{u}^{(\mathsf{rep})}_{\ell,i} &= (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}^{(\mathsf{rep})'}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i};\qquad \mathbf{u}_i &= (0^{N_i},\ 0^{N_i},\ r,\ 0,\ \rho_i,\ \mathbf{x}_i,\ 0^{N_i},\ r')_{\mathbf{B}_i}\\ \mathbf{v}^{(j)}_i &= (\mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \sigma_i,\ \pi^{(j)}_i,\ 0,\ \mathbf{y}^{(1,j)}_i,\ \mathbf{y}^{(0,j)}_i,\ \tau_i)_{\mathbf{B}^*_i};\quad \mathbf{v}^{(\mathsf{rep})}_{k,i} &= (\mathbf{y}^{(\mathsf{rep})}_{k,i},\mathbf{y}^{(\mathsf{rep})}_{k,i},\sigma_{k,i},\pi_{k,i},0,\mathbf{y}^{(\mathsf{rep})}_{k,i},\mathbf{y}^{(\mathsf{rep})}_{k,i},0)_{\mathbf{B}^*_i}\end{aligned}$$

where for each $j\in[J]$, $(\tau_i)_{i=1}^H$ 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 $\mathsf{G}_4,\mathsf{G}_5,\mathsf{G}_6$ is to swap $\mathbf{x}_i$ from coordinates $[2N_i+4,3N_i+3]$ to coordinates $[3N_i+4,4N_i+3]$ of $\mathbf{u}_i$, for all $i\in[H]$. The main idea is to consider the *selective* version G^*_j for $j\in\{4,5,6\}$, where the values $(\mathbf{x}_i[k],\mathbf{y}^{(1,j)}_i,\mathbf{y}^{(0,j)}_i)^{j\in[J]}_{i\in[H],k\in[N_i]}$ are guessed in advance. We then use formal argument for the transitions $\mathsf{G}^*_j\to\mathsf{G}^*_{j+1}$ for $j\in\{3,4,5\}$ to obtain

$$\Pr[\mathsf{G}^*_3=1]=\Pr[\mathsf{G}^*_4=1]=\Pr[\mathsf{G}^*_5=1]=\Pr[\mathsf{G}^*_6=1]\ . \tag{8}$$

In the end, we use a *complexity leveraging* argument to conclude that thanks to (8), we have $\Pr[\mathsf{G}_3=1]=\Pr[\mathsf{G}_4=1]=\Pr[\mathsf{G}_5=1]=\Pr[\mathsf{G}_6=1]$. For the sequence $\mathsf{G}_3\to\mathsf{G}_6$, we make a guess for the values $(\mathbf{x}_i[k],\mathbf{y}^{(1,j)}_i,\mathbf{y}^{(0,j)}_i)^{j\in[J]}_{i\in[H],k\in[N_i]}$, choose $r'\xleftarrow{\$}\mathbb{Z}_q^*$, random secret sharings $(\tau_i,\tilde{\tau}_i)_{i=1}^H$ of 0 for each $j\in[J]$, with $\tau_i,\tilde{\tau}_i\neq 0$ for all i, and define the event E that the guess is correct on $(\mathbf{x}_i[k],\mathbf{y}^{(1,j)}_i,\mathbf{y}^{(0,j)}_i)^{j\in[J]}_{i\in[H],k\in[N_i]}$ and

$$\tilde{\tau}_i-\tau_i=\frac{1}{r'}\langle\mathbf{x}_i,\Delta\mathbf{y}^{(j)}_i\rangle$$

where $\Delta\mathbf{y}^{(j)}_i := \mathbf{y}^{(1,j)}_i-\mathbf{y}^{(0,j)}_i$. We describe the selective games below, starting from G^*_3, where event E is assumed true:

Game G^*_3 : The selective version of G_3, assuming event E is true.

Game G_4^* : Knowing $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N_i]}^{j\in[J]}$ in advance, we perform a formal quotient on coordinates $[2N_i+4, 4N_i+3]$ of $\mathbf{u}_i$ and of $\mathbf{v}_i^{(j)}$. The bases are changed following:

$$B_i = \begin{cases} B_i[row,col] & = 1 \text{ if } row = col \leq 2N_i+3 \\ B_i[row,col] & = \frac{1}{\mathbf{x}_i[k]} \text{ if } \exists k \in [2N_i] : row = col = 2N_i+3+k \text{ AND } \mathbf{x}_i[k] \neq 0 \\ B_i[row,col] & = 1 \text{ if } \exists k \in [2N_i] : row = col = 2N_i+3+k \text{ AND } \mathbf{x}_i[k] = 0 \\ B_i[row,col] & = 1 \text{ if } row = col \geq 3N_i+3 \\ B_i[row,col] & = 0 \text{ otherwise} \end{cases} \tag{9}$$

$$B_i' := \left(B_i^{-1}\right)^\top; \mathbf{B}_i = B_i \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^* \ .$$

We note that the matrices, which have dimensions $(4N_i+4)\times(4N_i+4)$, depend only on i and not on j hence the basis change is well defined. The vectors change from $\mathbf{H}_i$ and $\mathbf{H}_i^*$ to $\mathbf{B}_i$ and $\mathbf{B}_i^*$ accordingly:

$$\begin{aligned} \mathbf{u}_{\ell,i}^{(\mathsf{rep})} &= (\cdots, 0, \cdots, 0, 0, \cdots, 0, 0)_{\mathbf{B}_i}; \mathbf{u}_i = (\cdots, 1, \cdots, 1, 0, \cdots, 0, r')_{\mathbf{B}_i} \\ \mathbf{v}_i^{(j)} &= (\cdots, \mathbf{x}_i[1]\mathbf{y}_i^{(1,j)}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_i^{(1,j)}[N_i], \mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_i^{(0,j)}[N_i], \tau_i)_{\mathbf{B}_i^*} \\ \mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\cdots, \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i], \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i], 0)_{\mathbf{B}_i^*} \end{aligned}$$

In summary, in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in G_3^*, else we are in G_4^*. The change is formal.

Game G_5^* : In this game we perform a formal basis change to move all the values 1 from coordinates $[2N_i+4, 3N_i+3]$ to coordinates $[3N_i+4, 4N_i+3]$ of $\mathbf{u}_i$. The bases are changed following:

$$B_i = \begin{cases} B_i[row,col] & = 1 \text{ if } row = col \\ B_i[row,col] & = \frac{1}{r'} \text{ if } \exists k \in [N_i] : (row,col) = (4N_i+4, 2N_i+3+k) \text{ AND } \mathbf{x}_i[k] \neq 0 \\ B_i[row,col] & = 0 \text{ if } \exists k \in [N_i] : (row,col) = (4N_i+4, 2N_i+3+k) \text{ AND } \mathbf{x}_i[k] = 0 \\ B_i[row,col] & = \frac{-1}{r'} \text{ if } \exists k \in [N_i] : (row,col) = (4N_i+4, 3N_i+3+k) \text{ AND } \mathbf{x}_i[k] \neq 0 \\ B_i[row,col] & = 0 \text{ if } \exists k \in [N_i] : (row,col) = (4N_i+4, 3N_i+3+k) \text{ AND } \mathbf{x}_i[k] = 0 \\ B_i[row,col] & = 0 \text{ otherwise} \end{cases} \tag{10}$$

$$B_i' := \left(B_i^{-1}\right)^\top; \mathbf{B}_i = B_i \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = B_i' \cdot \mathbf{H}_i^* \ .$$

We emphasize the the matrices are defined based on the knowledge of $(\mathbf{x}_i[k])_{i\in[H],k\in[N_i]}$ in this selective hybrid. The vectors change from $\mathbf{H}_i$ and $\mathbf{H}_i^*$ to $\mathbf{B}_i$ and $\mathbf{B}_i^*$ accordingly:

$$
\begin{aligned}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} &= (\cdots,0,\cdots,0,0,\cdots,0,0)_{\mathbf{B}_i}\\
\mathbf{u}_i &= (\cdots,1,\cdots,1,0,\cdots,0,r')_{\mathbf{H}_i}\\
&= (\cdots, \underbrace{1-1,\cdots,1-1}_{\text{only 1-coord is changed, 0-coord stays invariant}}, \overbrace{0+1,\cdots,0+1}^{\text{changed only if its } N_i\text{-th pred is 1}}, r')_{\mathbf{B}_i}\\
\mathbf{v}_i^{(j)} &= (\cdots,\mathbf{x}_i[1]\mathbf{y}_i^{(1,j)}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_i^{(1,j)}[N_i],\mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_i^{(0,j)}[N_i],\tau_i)_{\mathbf{H}_i^*}\\
&= (\cdots,\mathbf{x}_i[1]\mathbf{y}_i^{(1,j)}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_i^{(1,j)}[N_i],\mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_i^{(0,j)}[N_i],\\
&\qquad \tau_i+\frac{1}{r'}\sum_{\mathbf{x}_i[k]\neq 0}\mathbf{x}_i[k]\Delta\mathbf{y}^{(j)}[k])_{\mathbf{B}_i^*}\\
&= (\cdots,\mathbf{x}_i[1]\mathbf{y}_i^{(1,j)}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_i^{(1,j)}[N_i],\mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_i^{(0,j)}[N_i],\\
&\qquad \tau_i+\frac{1}{r'}\langle\mathbf{x}_i,\Delta\mathbf{y}^{(j)}\rangle)_{\mathbf{B}_i^*}\\
\mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\cdots,\mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i],\mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i],0)_{\mathbf{H}_i^*}\\
&= (\cdots,\mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i],\mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i],\\
&\qquad 0+\frac{1}{r'}\sum_{\mathbf{x}_i[k']\neq 0}\mathbf{x}_i[k'](\mathbf{y}_{k,i}^{(\mathsf{rep})}[k']-\mathbf{y}_{k,i}^{(\mathsf{rep})}[k']))_{\mathbf{B}_i^*}\\
&= (\cdots,\mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i],\mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1],\cdots,\mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i],0)_{\mathbf{B}_i^*}\ .
\end{aligned}
$$

In summary, in bases $(\mathbf{H}_i,\mathbf{H}_i^*)$ we have the vectors as in G_4^*, else we are in G_5^*. We emphasize that the secret sharings are updated to $\tilde{\tau}_i := \tau_i + \frac{1}{r'}\langle\mathbf{x}_i,\Delta\mathbf{y}_i^{(j)}\rangle$ and are stil independent of j thanks to the hypothesis that $\langle\mathbf{x}_i,\mathbf{y}_i^{(1,j)}-\mathbf{y}_i^{(0,j)}\rangle$ is a constant for all $j\in[J],i\in[H]$. The change is formal.

Game G_6^* : We perform once again a formal quotient as from G_3^* to G_4^*, on coordinates $[2N_i+4,4N_i+3]$ of $\mathbf{u}_i$ and of $\mathbf{v}_i^{(j)}$. The bases are changed following:

$$
B_i=\begin{cases}
B_i[row,col] &= 1 \text{ if } row=col\leq 2N_i+3\\
B_i[row,col] &= \frac{1}{\mathbf{x}_i[k]} \text{ if } \exists k\in[2N_i]: row=col=2N_i+3+k \text{ AND } \mathbf{x}_i[k]\neq 0\\
B_i[row,col] &= 1 \text{ if } \exists k\in[2N_i]: row=col=2N_i+3+k \text{ AND } \mathbf{x}_i[k]=0\\
B_i[row,col] &= 0 \text{ otherwise}
\end{cases}
$$

$$
B_i' := \left(B_i^{-1}\right)^\top; \mathbf{B}_i = B_i\cdot\mathbf{H}_i;\quad \mathbf{B}_i^* = B_i'\cdot\mathbf{H}_i^*\ .
$$

We note that the matrices, which have dimensions $(4N_i+4)\times(4N_i+4)$, depend only on i and not on j hence the basis change is well defined. The vectors change from $\mathbf{H}_i$ and $\mathbf{H}_i^*$ to $\mathbf{B}_i$ and

$\mathbf{B}_i^*$ accordingly:

$$\begin{aligned}
\mathbf{u}_{\ell,i}^{(\mathsf{rep})} &= (\cdots, 0, \cdots, 0, 0, \cdots, 0, 0)_{\mathbf{B}_i} \\
\mathbf{u}_i &= (\cdots, 0, \cdots, 0, 1, \cdots, 1, r')_{\mathbf{H}_i} = (\cdots, 0^{N_i}, \mathbf{x}_i, r')_{\mathbf{B}_i} \\
\mathbf{v}_i^{(j)} &= (\cdots, \mathbf{x}_i[1]\mathbf{y}_i^{(1,j)}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_i^{(1,j)}[N_i], \mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_i^{(0,j)}[N_i], \tau_i)_{\mathbf{H}_i^*} \\
&= (\cdots, \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)}, \tau_i)_{\mathbf{B}_i^*} \\
\mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\cdots, \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i], \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N_i]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N_i], 0)_{\mathbf{H}_i^*} \\
&= (\cdots, \mathbf{y}_{k,i}^{(\mathsf{rep})}, \mathbf{y}_{k,i}^{(\mathsf{rep})}, 0)_{\mathbf{B}_i^*} \ .
\end{aligned}$$

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.

The above games demonstrate relation (8). We now employ the complexity leveraging argument. Let us fix $j \in \{3,4,5\}$. For $t \in \{j, j+1\}$ let $\mathbf{Adv}_t(\mathcal{A}) := |\Pr[\mathsf{G}_t(\mathcal{A}) = 1] - 1/2|$ 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 $\mathbf{Adv}_t^*(\mathcal{B}^*) := |\Pr[\mathsf{G}_t^*(\mathcal{B}^*) = 1] - 1/2|$ 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 $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N_i]}^{j\in[J]}$, chooses $r' \xleftarrow{\$} \mathbb{Z}_q^*$, random secret sharings $(\tau_i, \tilde{\tau}_i)_{i=1}^H$ of 0 for each $j \in [J]$, with $\tau_i, \tilde{\tau}_i \neq 0$ for all i. Then $\mathcal{B}$ defines the event E that the guess is correct on $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N_i]}^{j\in[J]}$ and

$$\tilde{\tau}_i - \tau_i = \frac{1}{r'}\langle \mathbf{x}_i, \Delta\mathbf{y}_i^{(j)} \rangle$$

where $\Delta\mathbf{y}_i^{(j)} := \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}$. When $\mathcal{B}^*$ guesses successfully and E happens, 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 fixed probability γ and is independent of the view of $\mathcal{A}$, we have:[7]

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

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 $j \in \{3,4,5\}$, in particular $\Pr[\mathsf{G}_6 = 1] = \Pr[\mathsf{G}_3 = 1]$. After G_6, the vectors are now of the form:

$$\begin{aligned}
&\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}; \quad \mathbf{u}_i = (0,\ 0,\ r,\ 0,\ \rho_i,\ 0^{N_i},\ \mathbf{x}_i,\ r')_{\mathbf{B}_i} \\
&\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)}, \sigma_i, \pi_i^{(j)}, 0, \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)}, \tau_i')_{\mathbf{B}_i^*}; \quad \mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})}, \mathbf{y}_{k,i}^{(\mathsf{rep})}, \sigma_{k,i}, \pi_{k,i}, 0, \mathbf{y}_{k,i}^{(\mathsf{rep})}, \mathbf{y}_{k,i}^{(\mathsf{rep})}, 0)_{\mathbf{B}_i^*}.
\end{aligned}$$

We redo the computational swap from G_2 to G_3 so as to move $\mathbf{x}_i$ from coordinates $[3N_i + 4, 4N_i + 3]$ back to $[N_i + 1, 2N_i]$ of $\mathbf{u}_i$. The calculation is similar, using a sequence of $N_i + 1$ hybrids $\mathsf{G}_{6.k}$, namely $\mathsf{G}_{6.k}$ for $k \in [0, N_i]$, such that $\mathsf{G}_{6.0} = \mathsf{G}_6$ and in $\mathsf{G}_{6.k}$ the first coordinates $[3N_i + 3, 3N_i + 3 + k]$ are swapped to $[N_i + 1, N_i + k]$, for $k \geq 1$. For $k \in [N_i]$, the transition from $\mathsf{G}_{6.k-1}$ to $\mathsf{G}_{6.k}$ is described

[7] This calculation is the core of our complexity levaraging argument, being built upon the previous information-theoretic game transtions and the probability of event E.

below. 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:

$$\mathbf{B}_i = \begin{bmatrix} 1 & 0 & 0 \\ a & 1 & -a \\ 0 & 0 & 1 \end{bmatrix}_{N_i+k, 2N_i+3, 2N_i+4+k-1} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & -a & 0 \\ 0 & 1 & 0 \\ 0 & a & 1 \end{bmatrix}_{k, 2N_i+3, 3N_i+4+k-1} \cdot \mathbf{H}_i^* \ .$$

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

$$\begin{aligned}
\mathbf{u}_i &= (0^{N_i}, \underbrace{\mathbf{x}_i[1], .., \mathbf{x}_i[k-1], 0, .., 0}_{\text{last } (N_i-k+1)\text{-th coords are 0}}, \ r, \ 0, \ \rho_i, \ 0^{N_i}, \underbrace{0, .., 0, \mathbf{x}_i[k], .., \mathbf{x}_i[N_i]}_{\text{first } (k-1)\text{-th coords are 0}}, \ r')_{\mathbf{B}_i} \\
&\quad + (0^{N_i}, \underbrace{0, .., 0, c\mathbf{x}_i[k], 0, .., 0}_{k\text{-th coord among } N_i}, \ 0, \ 0, \ b\mathbf{x}_i[k], \ 0^{N_i}, \underbrace{0, .., 0, -c\mathbf{x}_i[k], 0, .., 0}_{k\text{-th coord among } N_i}, \ 0)_{\mathbf{H}_i} \\
&= (0^{N_i}, \underbrace{\mathbf{x}_i[1], .., \mathbf{x}_i[k-1], \boxed{\delta \mathbf{x}_i[k]}, .., 0}_{\text{last } (N_i-k)\text{-th coords are 0}}, \ r, \ 0, \ \boxed{\rho_i + b\mathbf{x}_i[k]}, \ 0^{N_i}, \underbrace{0, .., 0, \boxed{\mathbf{x}_i[k] - \delta \mathbf{x}_i[k]}, .., \mathbf{x}_i[N_i]}_{\text{first } (k-1)\text{-th coords are 0}}, \ r')_{\mathbf{B}_i}
\end{aligned}$$

$$\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'}, \ r_\ell, \ 0, \ \rho_{\ell,i}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i} \ .$$

We cannot compute $\mathbf{b}_{i,N_i+k}^*$ and $\mathbf{b}_{1,3N_i+4+k-1}^*$ due to the lack of $[\![a]\!]_2$, but the $\mathbf{v}$-vectors can be written in $\mathbf{H}_i^*$ indeed they stay invariant: for instance we consider $\mathbf{v}_i$, the same holds for $\mathbf{v}_{k,i}^{(\mathsf{rep})}$

$$\begin{aligned}
\mathbf{v}_i^{(j)} &= (\mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \sigma_i, \ \pi_i^{(j)}, \ 0, \ \mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \tau_i)_{\mathbf{H}_i^*} \\
&= (\mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \sigma_i, \ \pi_i^{(j)}, -a\mathbf{y}_i^{(0,j)}[k] + a\mathbf{y}_i^{(0,j)}[k], \ \mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \tau_i)_{\mathbf{B}_i^*} \\
&= (\mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \sigma_i, \ \pi_i^{(j)}, \ 0, \ \mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \tau_i)_{\mathbf{B}_i^*} \ .
\end{aligned}$$

If $\delta = 0$ we are in $\mathsf{G}_{6.k-1}$, else we are in $\mathsf{G}_{6.k}$, while updating ρ_i to $\rho_i + b\mathbf{x}_i[k]$ that stays uniformly random in $\mathbb{Z}_q$. We have $|\Pr[\mathsf{G}_{6.k} = 1] - \Pr[\mathsf{G}_{6.k-1} = 1]| \leq 2 \cdot \mathbf{Adv}_{\mathbb{G}_1}^{\mathsf{DDH}}(1^\lambda)$ and in the end $|\Pr[\mathsf{G}_7 = 1] - \Pr[\mathsf{G}_6 = 1]| \leq 2N_i \cdot \mathbf{Adv}_{\mathbb{G}_1}^{\mathsf{DDH}}(1^\lambda)$.
We redo the transition $\mathsf{G}_0 \to \mathsf{G}_1$ to clean coordinates $4N_i + 4$ of $\mathbf{u}_i, \mathbf{v}_i^{(j)}$, which leads to an additive loss $2 \cdot \mathbf{Adv}_{\mathbb{G}_2}^{\mathsf{DDH}}(1^\lambda) + 2 \cdot \mathbf{Adv}_{\mathbb{G}_1}^{\mathsf{DDH}}(1^\lambda)$. Then, we redo the transition $\mathsf{G}_1 \to \mathsf{G}_2$ to clean coordinates $[2N_i + 4, 3N_i + 3], [3N_i + 4, 4N_i + 3]$ of $\mathbf{v}_{k,i}^{(\mathsf{rep})}, \mathbf{v}_i^{(j)}$, which is formal. Finally we arrive at G_8 whose vectors are sampled according to D_1, implying $|\Pr[\mathsf{G}_8] - \Pr[\mathsf{G}_0]| \leq (2N_i + 8) \cdot \mathbf{Adv}_{\mathbb{G}_1, \mathbb{G}_2}^{\mathsf{SXDH}}(1^\lambda)$ and the proof is completed. □

B.4 Proof of Lemma 12 - Secret-Sharing Swapping with Repetitions

Lemma 12 (Secret Sharing Swapping with Repetitions). *Under the same hypotheses of Lemma 11, additionally with repetitions* $\mathbf{u}_i^{(\tilde{j})}$ *for* $\tilde{j} \in [J]$ *so that*

$$\sum_{i=1}^{H} \langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(0,j)} \rangle = \sum_{i=1}^{H} \langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)} \rangle \tag{6}$$

and for each $i \in [H]$, $\langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)} \rangle = c_i$ *being a constant for all* $\tilde{j}, j \in [J]$, *the following distributions are computationally indistinguishable under the* SXDH *assumption in* $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathbf{t}, g_1, g_2, g_\mathbf{t}, \mathbf{e}, q)$*:*

$$D_0 := \left\{ \begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'}, \ r_\ell, \ 0, \ \rho_{\ell,i}^{(\mathsf{rep})}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ (\mathbf{u}_i^{(\tilde{j})} = (\boxed{\mathbf{x}_i^{(\tilde{j})}}, \ \boxed{0^{N_i}}, \ r, \ 0, \ \rho_i^{(\tilde{j})}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H]}^{\tilde{j}\in[J]} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \sigma_i, \ \pi_i^{(j)}, \ 0, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})}, \ \mathbf{y}_{k,i}^{(\mathsf{rep})}, \ \sigma_{k,i}, \ \pi_{k,i}^{(\mathsf{rep})}, \ 0, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\};$$

$$D_1 := \left\{ \begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'}, \ r_\ell, \ 0, \ \rho_{\ell,i}^{(\mathsf{rep})}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ (\mathbf{u}_i^{(\tilde{j})} = (\boxed{0^{N_i}}, \ \boxed{\mathbf{x}_i^{(\tilde{j})}}, \ r, \ 0, \ \rho_i^{(\tilde{j})}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H]}^{\tilde{j}\in[J]} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \sigma_i, \ \pi_i^{(j)}, \ 0, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})}, \ \mathbf{y}_{k,i}^{(\mathsf{rep})}, \ \sigma_{k,i}, \ \pi_{k,i}^{(\mathsf{rep})}, \ 0, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \end{array} \right\}.$$

More specifically, we have:

$$\left| \Pr_{\mathsf{samp}\sim D_0}[\mathcal{A}(\mathsf{samp}) \to 1] - \Pr_{\mathsf{samp}\sim D_1}[\mathcal{A}(\mathsf{samp}) \to 1] \right| \leq (2N_i + 8) \cdot J \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

for any ppt $\mathcal{A}$ *with fixed* $\left(\mathbf{x}_i, (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}_{\ell,i}^{(\mathsf{rep})'})_{\ell=1}^L, R, \mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)}, (\mathbf{y}_{k,i}^{(\mathsf{rep})}, R_k)_{k=1}^K\right)_{i\in[H]}^{j\in[J]}$.

Proof (Of Lemma 12). We describe the games to change the vectors' distribution from D_0 to D_1 as follows.

Game G_0**:** The vectors are sampled according to D_0.

Game G_1**:** We perform a sequence of hyrbids $\mathsf{G}_{0.\tilde{j}}$ for $\tilde{j} \in [0, J]$ where $\mathsf{G}_{0.0} = \mathsf{G}_0$ and in the $\tilde{j}$-th hybrid we switch the first $\tilde{j}$ repetitions of $\mathbf{u}_i^{(\tilde{j})}$:

$$\begin{array}{l} (\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'}, \ r_\ell, \ 0, \ \rho_{\ell,i}^{(\mathsf{rep})}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\ (\mathbf{u}_i^{(j')} = (\mathbf{x}_i^{(j')}, \ 0^{N_i}, \ r, \ 0, \ \rho_i^{(j')}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H]} \text{ if } j' > \tilde{j} \\ (\mathbf{u}_i^{(j')} = (\boxed{0^{N_i}}, \ \boxed{\mathbf{x}_i^{(j')}}, \ r, \ 0, \ \rho_i^{(j')}, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i})_{i\in[H]} \text{ if } j' \leq \tilde{j} \\ (\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j)}, \ \mathbf{y}_i^{(0,j)}, \ \sigma_i, \ \pi_i^{(j)}, \ 0, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i^*})_{i\in[H]}^{j\in[J]} \\ (\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})}, \ \mathbf{y}_{k,i}^{(\mathsf{rep})}, \ \sigma_{k,i}, \ \pi_{k,i}^{(\mathsf{rep})}, \ 0, \ 0^{N_i}, \ 0^{N_i}, \ 0)_{\mathbf{B}_i^*})_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]} \ . \end{array}$$

This means $\mathsf{G}_{0.J}$ has all vectors sampled from D_1 and the proof is completed. For $\tilde{j} \in [J]$, to go from $\mathsf{G}_{0.\tilde{j}-1}$ to $\mathsf{G}_{0.\tilde{j}}$ we apply Lemma 11. This can be done in a black-box manner, where at each application we target only the $\tilde{j}$-th repetition of each $\mathbf{u}_i^{(\tilde{j})}$. We detail below how each transitions from the proof of Lemma 11 is used:

- $\underline{\mathsf{G}_{0.\tilde{j}-1.0}}$: This is $\mathsf{G}_{0.\tilde{j}-1}$.

- $\underline{\mathsf{G}_{0.\tilde{j}-1.1}}$: We use the same calculation as in $\mathsf{G}_0 \to \mathsf{G}_1$ in the proof of Lemma 11, noting that it is completely computational and we can modify only the $\tilde{j}$-th repetition:

$$\begin{aligned}
\big(\mathbf{u}_i^{(j')} &= (\mathbf{x}_i^{(j')},\ 0^{N_i},\ r,\ 0,\ \rho_i^{(j')},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\big)_{i\in[H]} \text{ if } j' > \tilde{j}\\
\big(\mathbf{u}_i^{(\tilde{j})} &= (\mathbf{x}_i^{(\tilde{j})},\ 0^{N_i},\ r,\ 0,\ \rho_i^{(\tilde{j})},\ 0^{N_i},\ 0^{N_i},\ \boxed{r'})_{\mathbf{B}_i}\big)_{i\in[H]}\\
\big(\mathbf{u}_i^{(j')} &= (0^{N_i},\ \mathbf{x}_i^{(j')},\ r,\ 0,\ \rho_i^{(j')},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\big)_{i\in[H]} \text{ if } j' \le \tilde{j}-1\\
\big(\mathbf{v}_i^{(j)} &= (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ \boxed{\tilde{\tau}_i})_{\mathbf{B}_i^*}\big)_{i\in[H]}^{j\in[J]}\\
\big(\mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\big)_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]}\ .
\end{aligned}$$

  where $r' \xleftarrow{\$} \mathbb{Z}_q^*$ and for all $j \in [J]$, the secret sharing $(\tau_i)_i$ in $\mathbf{v}$-vectors satisfies: $\sum_{i=1}^{H} \tau_i = 0$.
- $\underline{\mathsf{G}_{0.\tilde{j}-1.2}}$: We perform a formal duplication on *all* $\mathbf{v}$-vectors:

$$\begin{aligned}
\big(\mathbf{v}_i^{(j)} &= (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0, \boxed{\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)}},\ \tilde{\tau}_i)_{\mathbf{B}_i^*}\big)_{i\in[H]}^{j\in[J]}\\
\big(\mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0, \boxed{\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})}},\ 0)_{\mathbf{B}_i^*}\big)_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]}\ .
\end{aligned}$$

  This is done for all i and and all repetitions $\mathsf{rep} \in [J]$. Dually, the destination coordinates in the $\mathbf{u}$-vectors are all 0 hence they stay unchanged.
- $\underline{\mathsf{G}_{0.\tilde{j}-1.3}}$: We use a computational swap between $[1, N_i]$ and $[3N_i + 4, 4N_i + 3]$ in $\mathbf{u}_i^{(\tilde{j})}$ using $(2N_i + 3)$-randomness. It is important that the change is computational using DDH in $\mathbb{G}_1$, therefore we can target only the $\tilde{j}$-th repetition of $(\mathbf{u}_i^{(\tilde{j})})_i$. For all other $(\mathbf{u}_i^{(j')})_i$ where $j' \neq \tilde{j}$ their coordinates remain intact and are 0. The computation on the $\mathbf{v}$-vectors can be done similarly as from $\mathsf{G}_2 \to \mathsf{G}_3$ in the proof of Lemma 11.

$$\begin{aligned}
\big(\mathbf{u}_i^{(\tilde{j})} &= (\boxed{0^{N_i}},\ 0^{N_i},\ r,\ 0,\ \rho_i^{(j')}, \boxed{\mathbf{x}_i^{(\tilde{j})}},\ 0^{N_i},\ r')_{\mathbf{B}_i}\big)_{i\in[H]}\\
\big(\mathbf{v}_i^{(j)} &= (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ \mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \tilde{\tau}_i)_{\mathbf{B}_i^*}\big)_{i\in[H]}^{j\in[J]}\\
\big(\mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ 0)_{\mathbf{B}_i^*}\big)_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]}\ .
\end{aligned}$$

- $\underline{\mathsf{G}_{0.\tilde{j}-1.4}}$: We now perform the complexity leveraging argument on coordinates $[2N_i+4, 4N_i+4]$, in the same manner to $\mathsf{G}_3 \to \mathsf{G}_4 \to \mathsf{G}_5 \to \mathsf{G}_6$ in the proof of Lemma 11. The transitions are all formal and affect all vectors in both $\mathbf{B}_i$ and $\mathbf{B}_i^*$:
  - For the $\mathbf{u}$-vectors, only the $\tilde{j}$-th repetition has a non-zero $(4N_i + 4)$-th coordinate and the swapping will make change only to $(\mathbf{u}_i^{(\tilde{j})})_i$. Moreover, other quotient changes are on 0 coordinates for j'-th repetition whose $j' \neq \tilde{j}$ and incur no modifications.
  - For the $\mathbf{v}$-vectors, all quotient basis changes will modify their coordinates $[2N_i+4, 4N_i+4]$ with the *same* factor $(\mathbf{x}_i^{(\tilde{j})}[m])_m$. Later on, the formal swapping will modify the secret sharings τ_i into $\tau_i + \frac{1}{r'}\langle \mathbf{x}_i^{(\tilde{j})}, \Delta\mathbf{y}_i^{(j)} \rangle$, which stays a secret sharing of 0 and independent of j', j thanks to condition (6) as well as the fact that $\langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)} \rangle$ is constant for all j', j, for any fixed $i \in [H]$.

- $\underline{\mathsf{G}_{0.\tilde{j}-1.5} = \mathsf{G}_{0.\tilde{j}}}$: Finally we perform a computational swapping, after exiting the complexity leveraging, then cleaning by reversing the transitions $\mathsf{G}_{0.\tilde{j}-1.0} \to \mathsf{G}_{0.\tilde{j}-1.2}$:

$$
\begin{aligned}
(\mathbf{u}_{\ell,i}^{(\mathsf{rep})} &= (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}_{\ell,i}^{(\mathsf{rep})'},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i})_{i\in[H],\ell\in[L]}^{\mathsf{rep}\in[J]} \\
\big(\mathbf{u}_i^{(j')} &= (\mathbf{x}_i^{(j')},\ 0^{N_i},\ r,\ 0,\ \rho_i^{(j')},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\big)_{i\in[H]} \text{ if } j' > \tilde{j} \\
\big(\mathbf{u}_i^{(j')} &= (\boxed{0^{N_i}},\ \boxed{\mathbf{x}_i^{(j')}},\ r,\ 0,\ \rho_i^{(j')},\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i}\big)_{i\in[H]} \text{ if } j' \leq \tilde{j} \\
\big(\mathbf{v}_i^{(j)} &= (\mathbf{y}_i^{(1,j)},\ \mathbf{y}_i^{(0,j)},\ \sigma_i,\ \pi_i^{(j)},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\big)_{i\in[H]}^{j\in[J]} \\
\big(\mathbf{v}_{k,i}^{(\mathsf{rep})} &= (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{N_i},\ 0^{N_i},\ 0)_{\mathbf{B}_i^*}\big)_{i\in[H],k\in[K]}^{\mathsf{rep}\in[J]}\ .
\end{aligned}
$$

  The computational changes help us target the necessary vectors, while the formal (de-)duplication can be done without touching other $\mathbf{u}$-vectors as the affected coordinates are 0. We remark that the cleaning steps can be postponed until the very last $\mathsf{G}_{0.J-1.4} \to \mathsf{G}_{0.J-1.5} = \mathsf{G}_{0.J}$ to save redundant security loss, while changing the definition of the hybrids.

After arriving at $\mathsf{G}_{0.J} = \mathsf{G}_1$, the vectors are following D_1, the transitions are indistinguishable under SXDH, and the proof is finished. □

B.5 Details about our FH-DMCFE in Section 5

Security The security of our scheme is proven below.

Theorem 13. *The DMCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* adaptively one-challenge weakly 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 maximum number of distinct tags for ciphertexts to* $\mathcal{O}\mathsf{Enc}$, q_k *denote the maximum number of distinct tags for keys to* $\mathcal{O}\mathsf{KeyGen}$, J *denote the maximum number of repetitive challenge queries for possibly different messages, and* $\tilde{J}$ *denote the maximum number of repetitive challenge queries for possibly different keys. Then, for any* ppt *adversary* $\mathcal{A}$ *against* $\mathcal{E}$, *we have the following bound:*

$$
\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) \leq \Big(\Big((q_e+1)J + (q_k+1)\tilde{J}\Big)\cdot(2N_i+8) + q_k + q_e + 8N_i + 2\Big)\cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)
$$

Proof. The proof is done via a sequence of hybrid games. The games are depicted in Figure 5.

Game G_0: This is the experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}stat\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ of a ppt adversary $\mathcal{A}$, where $b \xleftarrow{\$} \{0,1\}$ is the challenge bit. Because we are in the *one-challenge* setting with *static corruption*, the adversary will declare since 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 $\mathsf{tag\text{-}f}_k \neq \mathsf{tag\text{-}f}^*$. For the ciphertext and key queries, challenge or not, the adversary can issue repetitions and we index the repetition by $j' \in [J]$ (respectively $\tilde{j}' \in [\tilde{J}]$) for the non-challenge and by $j \in [J]$ (respectively $\tilde{j} \in [\tilde{J}]$) for the challenge ciphertext (respectively key) components, where $J, \tilde{J}$ are maximum numbers of repetitions at any position $i \in [n]$ in ciphertext and key components in that order.
There are 2 secret sharings of 0, namely $(\tilde{s}_i)_i$ and $(\tilde{t}_i)_i$, that we generate from 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 KeyGen corresponding

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

$$
\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& \mathbf{x}_{\ell,i}^{(j')} & \omega_\ell & 0^{N_i} & 0^{N_i} & \tilde{t}_i\omega_\ell & 0 & \rho_{\ell,i}^{(j')} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i} & \tilde{s}_i\mu_k & 0^{N_i} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} = (& \mathbf{x}_i^{(b,j)} & \omega & 0^{N_i} & 0^{N_i} & \tilde{t}_i\omega & 0 & \rho_i^{(j)} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} = (& \mathbf{y}_i^{(b,\tilde{j})} & \tilde{s}_i\mu & 0^{N_i} & 0^{N_i} & \mu & \pi_i^{(\tilde{j})} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*}
\end{array}
$$

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)

$$
\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& \mathbf{x}_{\ell,i}^{(j')} & \omega_\ell & 0^{N_i} & 0^{N_i} & \boxed{t_{\ell,i}} & 0 & \rho_{\ell,i}^{(j')} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i} & \boxed{s_{k,i}} & 0^{N_i} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} = (& \mathbf{x}_i^{(b,j)} & \omega & 0^{N_i} & 0^{N_i} & \boxed{t_i} & 0 & \rho_i^{(j)} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} = (& \mathbf{y}_i^{(b,\tilde{j})} & \boxed{s_i} & 0^{N_i} & 0^{N_i} & \mu & \pi_i^{(\tilde{j})} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_2: (Subspace Indistinguishability)

$$
\begin{array}{rcccccccccc}
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i} & s_{k,i} & \boxed{\mathbf{y}_{k,i}} & 0^{N_i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_i^{(\tilde{j})} = (& \mathbf{y}_i^{(b,\tilde{j})} & s_i & \boxed{\mathbf{y}_i^{(1,\tilde{j})}} & 0^{N_i} & \mu & \pi_i^{(\tilde{j})} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_3: (Secret Sharing Swapping - Lemma 12 - over $\mathcal{O}\mathsf{Enc}$ tags)

$$
\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& \boxed{0^{N_i}} & \omega_\ell & \boxed{\mathbf{x}_{\ell,i}^{(j')}} & 0^{N_i} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{c}_i^{(j)} = (& \boxed{0^{N_i}} & \omega & \boxed{\mathbf{x}_i^{(b,j)}} & 0^{N_i} & t_i & 0 & \rho_i^{(j)} & 0^{2N_i+1} &)_{\mathbf{B}_i}
\end{array}
$$

Game G_4: (Subspace Indistinguishability)

$$
\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& 0^{N_i} & \omega_\ell & \mathbf{x}_{\ell,i}^{(j')} & \boxed{\mathbf{x}_{\ell,i}^{(j')}} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{c}_i^{(j)} = (& 0^{N_i} & \omega & \mathbf{x}_i^{(b,j)} & \boxed{\mathbf{x}_i^{(1,j)}} & t_i & 0 & \rho_i^{(j)} & 0^{2N_i+1} &)_{\mathbf{B}_i}
\end{array}
$$

Game G_5: (Secret Sharing Swapping - Lemma 12 - over $\mathcal{O}\mathsf{KeyGen}$ tags)

$$
\begin{array}{rcccccccccc}
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \mathbf{y}_{k,i} & s_{k,i} & \boxed{0^{N_i}} & \boxed{\mathbf{y}_{k,i}} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*} \\
\mathbf{d}_i^{(\tilde{j})} = (& \mathbf{y}_i^{(b,\tilde{j})} & s_i & \boxed{0^{N_i}} & \boxed{\mathbf{y}_i^{(1,\tilde{j})}} & \mu & \pi_i^{(\tilde{j})} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*}
\end{array}
$$

Game G_6: (Cleaning)

$$
\begin{array}{rcccccccccc}
\mathbf{c}_{\ell,i}^{(j')} = (& 0^{N_i} & \omega_\ell & \boxed{0^{N_i}} & \mathbf{x}_{\ell,i}^{(j')} & t_{\ell,i} & 0 & \rho_{\ell,i}^{(j')} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} = (& \boxed{0^{N_i}} & s_{k,i} & 0^{N_i} & \mathbf{y}_{k,i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} = (& 0^{N_i} & \omega & \boxed{0^{N_i}} & \mathbf{x}_i^{(1,j)} & t_i & 0 & \rho_i^{(j)} & 0^{2N_i+1} &)_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} = (& \boxed{0^{N_i}} & s_i & 0^{N_i} & \mathbf{y}_i^{(1,\tilde{j})} & \mu & \pi_i^{(\tilde{j})} & 0 & 0^{2N_i+1} &)_{\mathbf{B}_i^*}
\end{array}
$$

Fig. 5: Games for proving Theorem 13

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 the key and ciphertext components as follows: for the challenge bit b

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ \omega_\ell \tilde{t}_i,\ 0,\ \rho_{\ell,i}^{(j')}, 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ \mu_k \tilde{s}_i,\ 0^{N_i},\ 0^{N_i},\ \mu_k,\ \pi_{k,i},\ 0, 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} &= (\mathbf{x}_i^{(b,j)},\ \omega,\ 0^{N_i},\ 0^{N_i},\ \omega \tilde{t}_i,\ 0,\ \rho_i^{(j)}, 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} &= (\mathbf{y}_i^{(b,\tilde{j})},\ \mu \tilde{s}_i,\ 0^{N_i},\ 0^{N_i},\ \mu,\ \pi_i,\ 0, 0^{2N_i+1})_{\mathbf{B}_i^*}
\end{aligned}$$

We index by (j, j') (respectively $(\tilde{j}, \tilde{j}')$) the repetitions of challenge and non-challenge ciphertext components (respectively key components). Note that the admissibility condition in Definition 18 requires that $\mathbf{x}_i^{(0,j)} = \mathbf{x}_i^{(1,j)}$ (resp. $\mathbf{y}_i^0 = \mathbf{y}_i^1$) for all queries to $\mathcal{O}\mathsf{Enc}$ (resp. $\mathcal{O}\mathsf{DKeyGen}$) where $i \in \mathcal{C}$. 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.
In the following we define event $\mathsf{G}_i = 1$ to signify that "The output b' of $\mathcal{A}$ satisfies $b' = b$ in G_i". We have $\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n},\mathcal{A}}(1^\lambda) = |\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}|$ and a probability calculation shows that for two successive games $\mathsf{G}_{i-1}, \mathsf{G}_i$, $|\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i-1} = 1]|$ is the difference in probabilities that $\mathcal{A}$ outputs 1 in G_i versus that $\mathcal{A}$ outputs 1 in G_{i-1}. We now start the description of games.

Game G_1: The vectors are now:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ \omega_\ell \tilde{t}_i,\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ \mu_k \tilde{s}_i,\ 0^{N_i},\ 0^{N_i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} &= (\mathbf{x}_i^{(0,j)},\ \omega,\ 0^{N_i},\ 0^{N_i},\ \omega \tilde{t}_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} &= (\mathbf{y}_i^0,\ \mu \tilde{s}_i,\ 0^{N_i},\ 0^{N_i},\ \mu,\ \pi_i,\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}
\end{aligned}$$

In this game we replace the shifted secret shares of 0 in $\mathbf{d}_i^{(\tilde{j})}, \mathbf{d}_{k,i}^{(\tilde{j}')}$ (respectively $\mathbf{c}_i, \mathbf{c}_{\ell,i}^{(j')}$), 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 $\mathsf{H}_1(\mathsf{tag}) := [\![\omega]\!]_1, \mathsf{H}_1(\mathsf{tag\text{-}f}) := [\![\omega_\ell]\!]_1$. 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 $\mathbf{d}_i^{(\tilde{j})}, \mathbf{d}_{k,i}^{(\tilde{j}')}$ 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:

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

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 $k \in [0, q_k + 1]$ for changing the non-challenge key query $\mathbf{d}_{k,i}^{(\tilde{j}')}$ under $\mathsf{tag\text{-}f}_k$ as well as changing the challenge key query $\mathbf{d}_i^{(\tilde{j})}$ undef $\mathsf{tag\text{-}f}$.

In the hybrid $\mathsf{G}_{0.1.k}$ with $0 \leq k \leq q_k$, the first k non-challenge key queries $\mathbf{d}_{k,i}^{(\tilde{j}')}$ are having a random secret shares over $i \in \mathcal{H}$:

$$\mathbf{d}_{k,i}^{(\tilde{j}')} = (\mathbf{y}_{k,i}^{(\tilde{j}')},\ \boxed{s_{k,i}},\ 0^{N_i},\ 0^{N_i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}$$

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 $\mathbf{d}_i^{(\tilde{j})}$:

$$\mathbf{d}_i^{(\tilde{j})} = (\mathbf{y}_i^0,\ s_i,\ 0^{N_i},\ 0^{N_i},\ \mu,\ \pi_i,\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}$$

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 note that the secret shares are the same for all *repetitions* at position i under $\mathsf{tag\text{-}f}^*$, or $\mathsf{tag\text{-}f}_k$. 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 $(\llbracket a \rrbracket_2, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2)$ where $c - ab = 0$ or a uniformly random value. Given a ppt adversary $\mathcal{A}$ that can distinguish $\mathsf{G}_{0.1.k-1}$ 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 $\llbracket a \rrbracket_2$ 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 $\mu_k := a$.
- 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 $S_k := -\sum_{i\in\mathcal{C}} \tilde{s}_i$.
- Let us denote $H := |\mathcal{H}|$ 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 $\llbracket \mu_k \tilde{s}_i \rrbracket_2$ for responding to the k-th key query $\mathbf{d}_{k,i}^{(\tilde{j}')}$ (or the challenge $\mathbf{d}_i^{(\tilde{j})}$ 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 $b_{k,i} := \alpha_{k,i} b + \beta_{k,i}$, $c_{k,i} := \alpha_{k,i} c + \beta_{k,i} a$. We note that

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

  are efficiently computable from the DDH instance. Then, $\mathcal{B}$ uses $\llbracket c_{k,i} \rrbracket_2$ in the simulation of $\mathbf{d}_{k,i}^{(\tilde{j}')}$ (or $\mathbf{d}_i^{(\tilde{j})}$ in the last hybrid).
- Next, for the last H-th honest client, $\mathcal{B}$ computes and defines:

$$\llbracket c_{k,H} \rrbracket_2 := S_k \cdot \llbracket a \rrbracket_2 - \sum_{i\in\mathcal{H}\setminus\{H\}} \llbracket c_{k,i} \rrbracket_2 \tag{11}$$

  where S_k is known in clear from above and other honest $\llbracket c_{k,i} \rrbracket_2$ can be computed as explained. The adversary $\mathcal{B}$ then uses $\llbracket c_{k,H} \rrbracket_2$ to simulation the H-th key component of the k-th key query. We emphasize that we makes use of the *static* corruption 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 $\mathbf{d}_{k,i}^{(\tilde{j}')}[N_i + 1] = \mu_k \tilde{s}_i := a b_{k,i}$ and we are in $\mathsf{G}_{0.1.k-1}$; Else $\mathbf{d}_{k,i}^{(\tilde{j}')}[N_i + 1] = s_{k,i} := c_{k,i}$ is a totally uniformly random value such that $\sum_{i\in\mathcal{H}} c_{k,i} + \mu_k \sum_{i\in\mathcal{C}} \tilde{s}_i = a S_k + \mu_k \sum_{i\in\mathcal{C}} \tilde{s}_i = 0$ thanks to (11) and the definition of S_k.

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

$\underline{\mathsf{G}_{0.3}}$**:** We replace the shifted shares $\omega\tilde{t}_i, \omega_\ell\tilde{t}_i$ in $\mathbf{c}_i^{(j)}, \mathbf{c}_{\ell,i}^{(j')}$ by random secret shares $t_i, t_{\ell,i}$ for $i \in \mathcal{H}$, while preserving their sum. We recall that because multiple queries, even for the same $i \in [n]$, are authorized for the challenge ciphertext, the same $\omega\tilde{t}_i$ (replaced by t_i) will be used for all $\mathbf{c}_i^{(j)}$ for all j. The random secret shares $t_i, t_{\ell,i} \xleftarrow{\$} \mathbb{Z}_q$ 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:

$$\mathbf{c}_{\ell,i}^{(j')} = (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ \boxed{t_{\ell,i}},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i}; \quad \mathbf{d}_{k,i}^{(\tilde{j}')} = (\mathbf{y}_{k,i}^{(\tilde{j}')},\ \boxed{s_{k,i}},\ 0^{N_i},\ 0^{N_i},\ \mu_k,\ \pi_{k,i},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}$$

$$\mathbf{c}_i^{(j)} = (\mathbf{x}_i^{(0,j)},\ \omega,\ 0^{N_i},\ 0^{N_i},\ \boxed{t_i},\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i}; \quad \mathbf{d}_i^{(\tilde{j})} = (\mathbf{y}_i^0,\ \boxed{s_i},\ 0^{N_i},\ 0^{N_i},\ \mu,\ \pi_i,\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}$$

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 use DSDH in $\mathbb{G}_2$ to make $\mathbf{y}_{k,i}^{(\tilde{j}')}$ appear in coordinates $[N_i+2, 2N_i+1]$ of $\mathbf{d}^{(\tilde{j}')}$, as well as $\mathbf{y}_i^{(1,\tilde{j})}$ in coordinates $[N_i+2, 2N_i+1]$ of $\mathbf{d}_i^{(\tilde{j})}$.

We proceed by a sequence of N_i+1 hybrids, indexed by $m \in [0, N_i]$, such that the first hybrid is identical to G_1 and in the m-th hybrid the first coordinates $[N_i+2, N_i+1+m]$ of $\mathbf{d}_i^{(\tilde{j}')}, \mathbf{d}_i^{(\tilde{j})}$ are modified, for $m \geq 1$. For $m \in [N_i]$, the transition from the $(m-1)$-th hybrid to the m-th hybrid is described below. 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:

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

The bases $\mathbf{B}_i^*$ can be computed using $[\![a]\!]_2$ and the key components can be written as follows:

$$\begin{aligned}
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \underbrace{\mathbf{y}_{k,i}^{(\tilde{j}')}[1], .., \mathbf{y}_{k,i}^{(\tilde{j}')}[m-1], 0, .., 0}_{\text{last } (N_i-m+1)\text{-th coords are 0}},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0^{2N_i+2})_{\mathbf{B}_i^*} \\
&\quad + (0^{N_i+1},\ \underbrace{0, .., 0, c\mathbf{y}_{k,i}^{(\tilde{j}')}[m], 0, .., 0}_{m\text{-th coord among } N_i},\ 0^{N_i+1},\ b\mathbf{y}_{k,i}^{(\tilde{j}')}[m],\ 0^{2N_i+2})_{\mathbf{H}_i^*} \\
&= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \underbrace{\mathbf{y}_{k,i}^{(\tilde{j}')}[1], .., \mathbf{y}_{k,i}^{(\tilde{j}')}[m-1], \delta\mathbf{y}_{k,i}^{(\tilde{j}')}[m], .., 0}_{\text{last } (N_i-m)\text{-th coords are 0}},\ 0^{N_i},\ \mu_k, \pi_{k,i}^{(\tilde{j}')} + b\mathbf{y}_{k,i}^{(\tilde{j}')}[m],\ 0^{2N_i+2})_{\mathbf{B}_i^*} \\
\mathbf{d}_i^{(\tilde{j})} &= (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \underbrace{\mathbf{y}_i^{(1,\tilde{j})}[1], .., \mathbf{y}_i^{(1,\tilde{j})}[m-1], 0, .., 0}_{\text{last } (N_i-m+1)\text{-th coords are 0}},\ 0^{N_i},\ \mu,\ \pi_i^{(\tilde{j})},\ 0^{2N_i+2})_{\mathbf{B}_i^*} \\
&\quad + (0^{N_i+1},\ \underbrace{0, .., 0, c\mathbf{y}_i^{(1,\tilde{j})}[m], 0, .., 0}_{m\text{-th coord among } N_i},\ 0^{N_i+1},\ b\mathbf{y}_i^{(1,\tilde{j})}[m],\ 0^{2N_i+2})_{\mathbf{H}_i^*} \\
&= (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \underbrace{\mathbf{y}_i^{(1,\tilde{j})}[1], .., \mathbf{y}_i^{(1,\tilde{j})}[m-1], \delta\mathbf{y}_i^{(1,\tilde{j})}[m], .., 0}_{\text{last } (N_i-m)\text{-th coords are 0}},\ 0^{N_i},\ \mu, \pi_i^{(\tilde{j})} + b\mathbf{y}_i^{(1,\tilde{j})}[m],\ 0^{2N_i+2})_{\mathbf{B}_i^*} \ .
\end{aligned}$$

We update $\pi_{k,i}^{(\tilde{j}')}, \pi_i^{(\tilde{j})}$ to $\pi_{k,i}^{(\tilde{j}')} + b\mathbf{y}_{k,i}^{(\tilde{j}')}[m], \pi_i^{(\tilde{j})} + b\mathbf{y}_i^{(1,\tilde{j})}[m]$. Even though $\mathbf{b}_{i,N_i+1+m}$ cannot be computed due to the lack of $[\![a]\!]_1$, the simulator can write the $\mathbf{c}$-vectors in $\mathbf{H}_i$ to observe how they are affected:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{H}_i} \\
&= (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')} + 0 \cdot a,\ 0^{2N_i+1})_{\mathbf{B}_i} \\
&= (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{c}_i^{(j)} &= (\mathbf{x}_i^{(b,j)},\ \omega,\ 0^{N_i},\ 0^{N_i},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{H}_i} \\
&= (\mathbf{x}_i^{(b,j)},\ \omega,\ 0^{N_i},\ 0^{N_i},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i}\ .
\end{aligned}$$

If $\delta = 0$ we are in the $(m-1)$-th hybrid, else we are in the m-th hybrid. Totally, after N_i transitions we arrive at G_3 and obtain $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq 2N_i \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(1^\lambda)$.

Game G_3: After G_2 the vectors are now:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (\mathbf{x}_{\ell,i}^{(j')},\ \omega_\ell,\ 0^{N_i},\ 0^{N_i},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \boxed{\mathbf{y}_{k,i}^{(\tilde{j}')}},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} &= (\mathbf{x}_i^{(b,j)},\ \omega,\ 0^{N_i},\ 0^{N_i},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} &= (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \boxed{\mathbf{y}_i^{(1,\tilde{j})}},\ 0^{N_i},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}\ .
\end{aligned}$$

We now swap $\mathbf{x}_i^{(b,j)}, \mathbf{x}_{\ell,i}^{(j')}$ from coordinates $[1, N_i]$ to coordinates $[N_i + 2, 2N_i + 1]$ in $\mathbf{c}_i^{(j)}, \mathbf{c}_{\ell,i}^{(j')}$, respectively. This can be done by a sequence of $q_e + 2$ hybrids over the q_e distinct tags tag_ℓ to $\mathcal{O}\mathsf{Enc}$ and the only challenge tag tag^* that is declared at the beginning of the one-challenge game. The first hybrid is the same as G_3. The transition between each hybrid is done by an application of Lemma 12. We first swap the challenge $\mathbf{c}_i^{(j)}$, then swap the non-challenge $\mathbf{c}_{\ell,i}^{(j')}$ one after another on an ordering over ω_ℓ, *e.g.* their order of appearances. We verify the constraints required by Lemma 12.

Swapping the challenge $\mathbf{x}_i^{(b,j)}$:

- First of all, thanks to the weakly function-hiding admissibility (condition 2): for all $j \in [J], \tilde{j} \in [\tilde{J}]$

$$\sum_{i=1}^{H} \langle \mathbf{x}_i^{(b,j)}, \mathbf{y}_i^{(b,\tilde{j})} - \mathbf{y}_i^{(1,\tilde{j})} \rangle \overset{(1)}{=} \sum_{i=1}^{n} \langle \mathbf{x}_i^{(b,j)}, \mathbf{y}_i^{(b,\tilde{j})} - \mathbf{y}_i^{(1,\tilde{j})} \rangle \overset{(2)}{=} 0$$

  where (1) comes from condition 1 that for corrupted i the challenge keys satisfy $\mathbf{y}_i^{(0,\tilde{j})} = \mathbf{y}_i^{(1,\tilde{j})}$, and (2) comes from the weakly function-hiding.
- Moreover, equation (2) makes sure that for each $i \in H$, the quantity

$$\langle \mathbf{x}_i^{(b,j)}, \mathbf{y}_i^{(b,\tilde{j})} \rangle - \langle \mathbf{x}_i^{(b,j)}, \mathbf{y}_i^{(1,\tilde{j})} \rangle$$

  is a constant c_i for any $j \in [J], \tilde{j} \in [\tilde{J}]$.

The sets of vectors, listed in the order of the lemma's statement, are $((\mathbf{c}_{\ell,i}^{(j')})_{i\in\mathcal{H}}^{j'\in[J]}, (\mathbf{c}_i^{(j)})_{i\in\mathcal{H}}^{j\in[J]}, (\mathbf{d}_i^{(\tilde{j})})_{i\in\mathcal{H}}^{\tilde{j}\in[J]}, (\mathbf{d}_{k,i}^{(\tilde{j}')})_{i\in\mathcal{H},k\in[q_k]}^{\tilde{j}'\in[J]})$. The $4N_i + 4$ coordinates affected, in the order w.r.t the statement of Lemma 12 so that they form a subspace of dimension $4N_i + 4$, are $([1, N_i], [N_i + 2, 2N_i + 1], N_i + 1, 3N_i + 3, 3N_i + 4, [3N_i + 5, 5N_i + 5])$. In the end, the security loss for this swapping is bounded by $(2N_i + 8) \cdot J \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

Swapping the non-challenge $\mathbf{x}_{\ell,i}^{(j')}$:

- First of all, thanks to the weakly function-hiding admissibility (condition 2): for all $j' \in [J], \tilde{j} \in [\tilde{J}]$

$$\sum_{i=1}^{H} \langle \mathbf{x}_{\ell,i}^{(j')}, \mathbf{y}_i^{(b,\tilde{j})} - \mathbf{y}_i^{(1,\tilde{j})} \rangle \overset{(1)}{=} \sum_{i=1}^{n} \langle \mathbf{x}_{\ell,i}^{(j')}, \mathbf{y}_i^{(b,\tilde{j})} - \mathbf{y}_i^{(1,\tilde{j})} \rangle \overset{(2)}{=} 0$$

  where (1) comes from condition 1 that for corrupted i the challenge keys satisfy $\mathbf{y}_i^{(0,\tilde{j})} = \mathbf{y}_i^{(1,\tilde{j})}$, and (2) comes from the weakly function-hiding while treating $\mathbf{x}_{\ell,i}^{(j')}$ as the challenge.

- Moreover, equation (2) makes sure that for each $i \in \mathcal{H}$, the quantity

$$\langle \mathbf{x}_i^{(j')}, \mathbf{y}_i^{(b,\tilde{j})} \rangle - \langle \mathbf{x}_i^{(j')}, \mathbf{y}_i^{(1,\tilde{j})} \rangle$$

  is a constant c_i for any $j' \in [J], \tilde{j} \in [\tilde{J}]$.

The sets of vectors, listed in the order of the lemma's statement, are $((\mathbf{c}_i^{(j)})_{i\in\mathcal{H}}^{j\in[J]}, (\mathbf{c}_{\ell,i}^{(j')})_{i\in\mathcal{H}}^{j'\in[J]}, (\mathbf{d}_i^{(\tilde{j})})_{i\in\mathcal{H}}^{\tilde{j}\in[J]}, (\mathbf{d}_{k,i}^{(\tilde{j}')})_{i\in\mathcal{H},k\in[q_k]}^{\tilde{j}'\in[J]})$. The $4N_i + 4$ coordinates affected, in the order w.r.t the statement of Lemma 12 so that they form a subspace of dimension $4N_i + 4$, are $([1, N_i], [N_i + 2, 2N_i + 1], N_i + 1, 3N_i + 3, 3N_i + 4, [3N_i + 5, 5N_i + 5])$. Finally, the security loss for each swap over the q_e non-challenge tags to $\mathcal{O}\mathsf{Enc}$ is upper bounded by: $(2N_i + 8) \cdot J \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$. In total, we have $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \leq (q_e + 1) \cdot (2N_i + 8) \cdot J \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

Game G_4: After G_3 the vectors are now:

$$\mathbf{c}_{\ell,i}^{(j')} = (\boxed{0^{N_i}},\ \omega_\ell,\ \boxed{\mathbf{x}_{\ell,i}^{(j')}},\ 0^{N_i},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0...)_{\mathbf{B}_i}; \quad \mathbf{c}_i^{(j)} = (\boxed{0^{N_i}},\ \omega,\ \boxed{\mathbf{x}_i^{(b,j)}},\ 0^{N_i},\ t_i,\ 0,\ \rho_i^{(j)},\ 0...)_{\mathbf{B}_i}$$
$$\mathbf{d}_{k,i}^{(\tilde{j}')} = (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0...)_{\mathbf{B}_i^*}; \quad \mathbf{d}_i^{(\tilde{j})} = (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \mathbf{y}_i^{(1,\tilde{j})},\ 0^{N_i},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0...)_{\mathbf{B}_i^*}\ .$$

In this G_4, we use DSDH to make $\mathbf{x}_{\ell,i}^{(\tilde{j})}$ appear in coordinates $[2N_i + 2, 3N_i + 1]$ of $\mathbf{c}_i^{(j')}$, as well as $\mathbf{x}_i^{(1,\tilde{j})}$ in coordinates $[2N_i + 2, 3N_i + 1]$ of $\mathbf{c}_i^{(j)}$.

We proceed by a sequence of $N_i + 1$ hybrids, indexed by $m \in [0, N_i]$, such that the first hybrid is identical to G_3 and in the m-th hybrid the first coordinates $[2N_i + 2, 2N_i + 1 + m]$ of $\mathbf{d}^{(\tilde{j}')}, \mathbf{d}_i^{(\tilde{j})}$ are modified, for $m \geq 1$. For $m \in [N_i]$, the transition from the $(m-1)$-th hybrid to the m-th hybrid is described below. 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:

$$\mathbf{B}_i = \begin{bmatrix} 1 & \\ a & 1 \end{bmatrix}_{2N_i+1+m,3N_i+4} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}_{2N_i+1+m,3N_i+4} \cdot \mathbf{H}_i^*\ .$$

The basis $\mathbf{B}_i$ can be computed using $[\![a]\!]_1$ and the $\mathbf{c}$-vectors are simulated below:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \underbrace{\mathbf{x}_{\ell,i}^{(j')}[1],..,\mathbf{x}_{\ell,i}^{(j')}[m-1],0,..,0}_{\text{last } (N_i-m+1)\text{-th coords are 0}},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
&\quad + (0^{2N_i+1},\ \underbrace{0,..,0,c\mathbf{x}_{\ell,i}^{(j')}[m],0,..,0}_{m\text{-th coord among } N_i},\ 0^2,\ b\mathbf{x}_{\ell,i}^{(j')}[m],\ 0^{2N_i+2})_{\mathbf{H}_i} \\
&= (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \underbrace{\mathbf{x}_{\ell,i}^{(j')}[1],..,\mathbf{x}_{\ell,i}^{(j')}[m-1],\delta\mathbf{x}_{\ell,i}^{(j')}[m],..,0}_{\text{last } (N_i-m)\text{-th coords are 0}},\ t_{\ell,i},0,\ \rho_{\ell,i}^{(j')}+b\mathbf{x}_{\ell,i}^{(j')}[m],\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{c}_i^{(j)} &= (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \underbrace{\mathbf{x}_i^{(1,j)}[1],..,\mathbf{x}_i^{(1,j)}[m-1],0,..,0}_{\text{last } (N_i-m+1)\text{-th coords are 0}},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
&\quad + (0^{2N_i+1},\ \underbrace{0,..,0,c\mathbf{x}_i^{(1,j)}[m],0,..,0}_{m\text{-th coord among } N_i},\ 0^2,\ b\mathbf{x}_i^{(1,j)}[m],\ 0^{2N_i+2})_{\mathbf{H}_i} \\
&= (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \underbrace{\mathbf{x}_i^{(1,j)}[1],..,\mathbf{x}_i^{(1,j)}[m-1],\delta\mathbf{x}_i^{(1,j)}[m],..,0}_{\text{last } (N_i-m+1)\text{-th coords are 0}},\ t_i,0,\ \rho_i^{(j)}+b\mathbf{x}_i^{(1,j)}[m],\ 0^{2N_i+1})_{\mathbf{B}_i}\ .
\end{aligned}$$

Even though we cannot compute $\mathbf{b}^*_{i,2N_i+1+m}$ due to the lack of $[\![a]\!]_2$, the $\mathbf{d}$-vectors can be written directly in $\mathbf{H}_i^*$:

$$\begin{aligned}
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0^{2N_i+1})_{\mathbf{H}_i^*} \\
&= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0+a\cdot 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
&= (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{d}_i^{(\tilde{j})} &= (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \mathbf{y}_i^{(1,\tilde{j})},\ 0^{N_i},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0^{2N_i+1})_{\mathbf{H}_i^*} \\
&= (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \mathbf{y}_i^{1},\ 0^{N_i},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}\ .
\end{aligned}$$

If $\delta = 0$ we are in the $(m-1)$-th hybrid, else we are in the m-th hybrid. Totally, after N_i transitions we arrive at G_5 and obtain $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq 2N_i \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

Game G_5: After G_4 the vectors are now:

$$\begin{aligned}
&\mathbf{c}_{\ell,i}^{(j')} = (\boxed{0^{N_i}},\ \omega_\ell,\ \boxed{\mathbf{x}_{\ell,i}^{(j')}},\ \boxed{\mathbf{x}_{\ell,i}^{(j')}},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0...)_{\mathbf{B}_i};\quad \mathbf{c}_i^{(j)} = (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \boxed{\mathbf{x}_i^{(1,j)}},\ t_i,\ 0,\ \rho_i^{(j)},\ 0...)_{\mathbf{B}_i} \\
&\mathbf{d}_{k,i}^{(\tilde{j}')} = (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ 0^{N_i},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0...)_{\mathbf{B}_i^*};\quad \mathbf{d}_i^{(\tilde{j})} = (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \mathbf{y}_i^{(1,\tilde{j})},\ 0^{N_i},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0...)_{\mathbf{B}_i^*}
\end{aligned}$$

We apply Lemma 12 to swap $\mathbf{y}_i^{(1,\tilde{j})}, \mathbf{y}_{k,i}^{(\tilde{j}')}$ from coordinates $[N_i+2, 2N_i+1]$ to coordinates $[2N_i+2, 3N_i+1]$ of vectors $\mathbf{d}_i^{(\tilde{j})}, \mathbf{d}_{k,i}^{(\tilde{j}')}$. This can be done by a sequence of q_k+2 hybrids over the q_k distinct tags $\mathsf{tag\text{-}f}_k$ to $\mathcal{O}\mathsf{KeyGen}$ and the only challenge tag $\mathsf{tag\text{-}f}$ that is declared at the beginning of the one-challenge game. The first hybrid is the same as G_5. The transition between each hybrid is done by an application of Lemma 12. We first swap the challenge $\mathbf{d}_i^{(\tilde{j})}$, then swap the non-challenge $\mathbf{d}_{k,i}^{(\tilde{j}')}$ one after another on an ordering over μ_k, *e.g.* their order of appearances. We verify the constraints required by Lemma 12.

Swapping the challenge $\mathbf{y}_i^{(1,\tilde{j})}$:

- First of all, thanks to the weakly function-hiding admissibility (condition 2): for all $j \in [J], \tilde{j} \in [\tilde{J}]$

$$\sum_{i=1}^{H} \langle \mathbf{y}_i^{(1,\tilde{j})}, \mathbf{x}_i^{(b,j)} - \mathbf{x}_i^{(1,j)} \rangle \overset{(1)}{=} \sum_{i=1}^{n} \langle \mathbf{y}_i^{(1,\tilde{j})}, \mathbf{x}_i^{(b,j)} - \mathbf{x}_i^{(1,j)} \rangle \overset{(2)}{=} 0$$

where (1) comes from condition 1 that for corrupted i the challenge messages satisfy $\mathbf{x}_i^{(b,j)} = \mathbf{x}_i^{(1,j)}$, and (2) comes from the weakly function-hiding.

- Moreover, equation (2) makes sure that for each $i \in H$, the quantity

$$\langle \mathbf{y}_i^{(1,\tilde{j})}, \mathbf{x}_i^{(b,j)} \rangle - \langle \mathbf{y}_i^{(1,\tilde{j})}, \mathbf{x}_i^{(1,j)} \rangle$$

is a constant c_i for any $j \in [J], \tilde{j} \in [\tilde{J}]$.

The sets of vectors, listed in the order of the lemma's statement, are $((\mathbf{d}_{k,i}^{(\tilde{j}')})_{i\in\mathcal{H},k\in[q_k]}^{\tilde{j}'\in[J]}, (\mathbf{d}_i^{(\tilde{j})})_{i\in\mathcal{H}}^{\tilde{j}\in[J]}, (\mathbf{c}_i^{(j)})_{i\in\mathcal{H}}^{j\in[J]}, (\mathbf{c}_{\ell,i}^{(j')})_{i\in\mathcal{H}}^{j'\in[J]})$. The $4N_i + 4$ coordinates affected, in the order w.r.t the statement of Lemma 12 so that they form a subspace of dimension $4N_i + 4$, are $([N_i + 2, 2N_i + 1], [2N_i + 2, 3N_i + 1], 3N_i + 2, 3N_i + 3, 3N_i + 4, [3N_i + 5, 5N_i + 5])$. Finally this swap incurs a security loss upper bounded by $(2N_i + 8) \cdot \tilde{J} \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

Swapping the non-challenge $\mathbf{y}_{k,i}^{(\tilde{j}')}$:

- First of all, thanks to the weakly function-hiding admissibility (condition 2): for all $j \in [J], \tilde{j}' \in [\tilde{J}]$

$$\sum_{i=1}^{H} \langle \mathbf{y}_{k,i}^{(\tilde{j}')}, \mathbf{x}_i^{(b,j)} - \mathbf{x}_i^{(1,j)} \rangle \overset{(1)}{=} \sum_{i=1}^{n} \langle \mathbf{y}_{k,i}^{(\tilde{j}')}, \mathbf{x}_i^{(b,j)} - \mathbf{x}_i^{(1,j)} \rangle \overset{(2)}{=} 0$$

where (1) comes from condition 1 that for corrupted i the challenge messages satisfy $\mathbf{x}_i^{(b,j)} = \mathbf{x}_i^{(1,j)}$, and (2) comes from the weakly function-hiding.

- Moreover, equation (2) makes sure that for each $i \in H$, the quantity

$$\langle \mathbf{y}_i^{(1,\tilde{j})}, \mathbf{x}_i^{(b,j)} \rangle - \langle \mathbf{y}_i^{(1,\tilde{j})}, \mathbf{x}_i^{(1,j)} \rangle$$

is a constant c_i for any $j \in [J], \tilde{j} \in [\tilde{J}]$.

The sets of vectors, listed in the order of the lemma's statement, are $((\mathbf{d}_i^{(\tilde{j})})_{i\in\mathcal{H}}^{\tilde{j}\in[J]}, (\mathbf{d}_{k,i}^{(\tilde{j}')})_{i\in\mathcal{H},k\in[q_k]}^{\tilde{j}'\in[J]}, (\mathbf{c}_i^{(j)})_{i\in\mathcal{H}}^{j\in[J]}, (\mathbf{c}_{\ell,i}^{(j')})_{i\in\mathcal{H}}^{j'\in[J]})$. The $4N_i + 4$ coordinates affected, in the order w.r.t the statement of Lemma 12 so that they form a subspace of dimension $4N_i + 4$, are $([N_i + 2, 2N_i + 1], [2N_i + 2, 3N_i + 1], 3N_i + 2, 3N_i + 3, 3N_i + 4, [3N_i + 5, 5N_i + 5])$. Finally, the security loss for each swap over the q_e non-challenge tags to $\mathcal{O}\mathsf{Enc}$ is upper bounded by: $(2N_i + 8) \cdot \tilde{J} \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$. In total, we have $|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq (q_k + 1) \cdot (2N_i + 8) \cdot \tilde{J} \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

Game G_6: After G_5 the vectors are now:

$$\mathbf{c}_{\ell,i}^{(j')} = (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0...)_{\mathbf{B}_i}; \qquad \mathbf{c}_i^{(j)} = (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0,\ \rho_i^{(j)},\ 0...)_{\mathbf{B}_i}$$

$$\mathbf{d}_{k,i}^{(\tilde{j}')} = (\mathbf{y}_{k,i}^{(\tilde{j}')},\ s_{k,i},\ \boxed{0^{N_i}},\ \boxed{\mathbf{y}_{k,i}^{(\tilde{j}')}},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0...)_{\mathbf{B}_i^*}; \quad \mathbf{d}_i^{(\tilde{j})} = (\mathbf{y}_i^{(b,\tilde{j})},\ s_i,\ \boxed{0^{N_i}},\ \boxed{\mathbf{y}_i^{(1,\tilde{j})}},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0...)_{\mathbf{B}_i^*}\ .$$

We perform some cleanings to make the vectors independent of b. This is done in two steps:

$\underline{\mathsf{G}_{5.1}}$: We use DSDH in $\mathbb{G}_2$ to clean $\mathbf{y}_i^{(\tilde{j}')}, \mathbf{y}_i^{(b,\tilde{j})}$ at coordinates $[1, N_i]$ of vectors $\mathbf{d}_i^{(\tilde{j}')}, \mathbf{d}_i^{(\tilde{j})}$ respectively.

We proceed by a sequence of $N_i + 1$ hybrids, indexed by $m \in [0, N_i]$, such that the first hybrid is identical to G_5 and in the m-th hybrid the first m coordinates $[1, m]$ of $\mathbf{d}_i^{(\tilde{j}')}, \mathbf{d}_i^{(\tilde{j})}$ are cleaned, for $m \geq 1$. For $m \in [N_i]$, the transition from the $(m-1)$-th hybrid to the m-th hybrid is described below. Given a DSDH instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta := c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

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

The bases $\mathbf{B}_i^*$ can be computed using $[\![a]\!]_2$ and the key components can be written as follows:

$$\begin{aligned}
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\underbrace{0,..,0,\mathbf{y}_{k,i}^{(\tilde{j}')}[m],..,\mathbf{y}_{k,i}^{(\tilde{j}')}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ s_{k,i},\ 0^{N_i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
&\quad + (\underbrace{0,..,0,c\mathbf{y}_{k,i}^{(\tilde{j}')}[m],0,..,0}_{m\text{-th coord among } N_i},0^{2N_i+2},b\mathbf{y}_{k,i}^{(\tilde{j}')}[m],\ 0^{N_i+2})_{\mathbf{H}_i^*} \\
&= (\underbrace{0,..,0,(1-\delta)\mathbf{y}_{k,i}^{(\tilde{j}')}[m],..,\mathbf{y}_{k,i}^{(\tilde{j}')}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ s_{k,i},\ 0^{N_i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')}+b\mathbf{y}_{k,i}^{(\tilde{j}')}[m],\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{d}_i^{(\tilde{j})} &= (\underbrace{0,..,0,\mathbf{y}_i^{(b,\tilde{j})}[m],..,\mathbf{y}_i^{(b,\tilde{j})}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ s_i,\ 0^{N_i},\ \mathbf{y}_i^{(1,\tilde{j})},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
&\quad + (\underbrace{0,..,0,c\mathbf{y}_i^{(b,\tilde{j})}[m],0,..,0}_{m\text{-th coord among } N_i},0^{2N_i+2},b\mathbf{y}_i^{(b,\tilde{j})}[m],\ 0^{N_i+2})_{\mathbf{H}_i^*} \\
&= (\underbrace{0,..,0,(1-\delta)\mathbf{y}_i^{(b,\tilde{j})}[m],..,\mathbf{y}_i^{(b,\tilde{j})}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ s_i,\ 0^{N_i},\ \mathbf{y}_i^{(1,\tilde{j})},\ \mu,\ \pi_i^{(\tilde{j})}+\mathbf{y}_i^{(b,\tilde{j})}[m],\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}\ .
\end{aligned}$$

We are updating $\pi_{k,i}^{(\tilde{j}')}, \pi_i^{(\tilde{j})}$ to $\pi_{k,i}^{(\tilde{j}')} + b\mathbf{y}_{k,i}^{(\tilde{j}')}[m], \pi_i^{(\tilde{j})} + \mathbf{y}_i^{(b,\tilde{j})}[m]$.
We cannot compute $\mathbf{b}_{i,3N_i+6}$ due to the lack of $[\![a]\!]_1$, but the $\mathbf{c}$-vectors can be written in $\mathbf{H}_i$ and the changes does not affect them into $\mathbf{B}_i$:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{H}_i} \\
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0-a\cdot 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{c}_i^{(j)} &= (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{H}_i} \\
\mathbf{c}_i^{(j)} &= (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i}\ .
\end{aligned}$$

If $\delta = 0$ we are in the $(m-1)$-th hybrid, else we are cleaning coordinate m and in the m-th hybrid. After N_i transitions we clean all N_i coordinates and obtain $|\Pr[\mathsf{G}_{5.1}=1] - \Pr[\mathsf{G}_5 = 1]| \leq 2N_i \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

$\underline{\mathsf{G}_{5.2} = \mathsf{G}_6}$: After $\mathsf{G}_{5.1}$ the vectors are now:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \mathbf{x}_{\ell,i}^{(j')},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\boxed{0^{N_i}},\ s_{k,i},\ 0^{N_i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} &= (0^{N_i},\ \omega,\ \mathbf{x}_i^{(b,j)},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} &= (\boxed{0^{N_i}},\ s_i,\ 0^{N_i},\ \mathbf{y}_i^{(1,\tilde{j})},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}\ .
\end{aligned}$$

We now use DSDH to clean $\mathbf{x}_{\ell,i}^{(\tilde{j})}, \mathbf{x}_i^{(b,\tilde{j})}$ at coordinates $[N_i+2, 3N_i+1]$ of $\mathbf{c}_i^{(j')}, \mathbf{c}_i^{(j)}$.
We proceed by a sequence of N_i+1 hybrids, indexed by $m \in [0, N_i]$, such that the first hybrid is identical to $\mathsf{G}_{5.1}$ and in the m-th hybrid the first m coordinates $[N_i+2, N_i+1+m]$ of $\mathbf{c}_i^{(j')}, \mathbf{c}_i^{(j)}$ are cleaned, for $m \geq 1$. For $m \in [N_i]$, the transition from the $(m-1)$-th hybrid to the m-th hybrid is described below. 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:

$$\mathbf{B}_i = \begin{bmatrix} 1 & \\ a & 1 \end{bmatrix}_{N_i+1+m, 3N_i+4} \cdot \mathbf{H}_i; \quad \mathbf{B}_i^* = \begin{bmatrix} 1 & -a \\ 0 & 1 \end{bmatrix}_{N_i+1+m, 3N_i+4} \cdot \mathbf{H}_i^* \ .$$

The basis $\mathbf{B}_i$ can be computed using $[\![a]\!]_1$ and the ciphertext components can be written as follows:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \underbrace{0,..,0,\mathbf{x}_{\ell,i}^{(j')}[m],..,\mathbf{x}_{\ell,i}^{(j')}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
&\quad + (0^{N+i+1}, \underbrace{0,..,0,c\mathbf{x}_{\ell,i}^{(j')}[m],0,..,0}_{m\text{-th coord among } N_i}, 0^{N_i+2}, b\mathbf{x}_{\ell,i}^{(j')}[m], 0^{2N_i+1})_{\mathbf{H}_i} \\
&= (0^{N_i},\ \omega_\ell,\ \underbrace{0,..,0,(1-\delta)\mathbf{x}_{\ell,i}^{(j')}[m],..,\mathbf{x}_{\ell,i}^{(j')}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0, \rho_{\ell,i}^{(j')} + b\mathbf{x}_{\ell,i}^{(j')}[m],\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{c}_i^{(j)} &= (0^{N_i},\ \omega,\ \underbrace{0,..,0,\mathbf{x}_i^{(b,j)}[m],..,\mathbf{x}_i^{(b,j)}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
&\quad + (0^{N+i+1}, \underbrace{0,..,0,c\mathbf{x}_i^{(b,j)}[m],0,..,0}_{m\text{-th coord among } N_i}, 0^{N_i+2}, b\mathbf{x}_i^{(b,j)}[m], 0^{2N_i+1})_{\mathbf{H}_i} \\
&= (0^{N_i},\ \omega,\ \underbrace{0,..,0,(1-\delta)\mathbf{x}_i^{(b,j)}[m],..,\mathbf{x}_i^{(b,j)}[N_i]}_{\text{first } (m-1)\text{-th coords are 0}},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0, \rho_i^{(j)} + b\mathbf{x}_i^{(b,j)}[m],\ 0^{2N_i+1})_{\mathbf{B}_i} \ .
\end{aligned}$$

Even though we cannot compute $\mathbf{b}_{i,N_i+1+m}^*$ due to the lack of $[\![a]\!]_2$, similar to the transition $\mathsf{G}_5 \to \mathsf{G}_{5.1}$, the $\mathbf{d}$-vectors can be written in $\mathbf{H}_i^*$ and the basis changes does not modify them when going into $\mathbf{B}_i^*$. This is because $\mathbf{d}_{k,i}^{(\tilde{j}')}[N_i+1+m] = \mathbf{d}_i^{(\tilde{j})}[N_i+1+m] = 0$. If $\delta = 0$ we are in the $(m-1)$-th hybrid, else we are cleaning coordinate m and in the m-th hybrid. After N_i transitions we clean all N_i coordinates and obtain $|\Pr[\mathsf{G}_{5.2} = 1] - \Pr[\mathsf{G}_{5.1} = 1]| \leq 2N_i \cdot \mathbf{Adv}_{\mathbb{G}_1,\mathbb{G}_2}^{\mathsf{SXDH}}(1^\lambda)$.

In the end, we clean the coordinates and the vectors become:

$$\begin{aligned}
\mathbf{c}_{\ell,i}^{(j')} &= (0^{N_i},\ \omega_\ell,\ \boxed{0^{N_i}},\ \mathbf{x}_{\ell,i}^{(j')},\ t_{\ell,i},\ 0,\ \rho_{\ell,i}^{(j')},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_{k,i}^{(\tilde{j}')} &= (\boxed{0^{N_i}},\ s_{k,i},\ 0^{N_i},\ \mathbf{y}_{k,i}^{(\tilde{j}')},\ \mu_k,\ \pi_{k,i}^{(\tilde{j}')},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*} \\
\mathbf{c}_i^{(j)} &= (0^{N_i},\ \omega,\ \boxed{0^{N_i}},\ \mathbf{x}_i^{(1,j)},\ t_i,\ 0,\ \rho_i^{(j)},\ 0^{2N_i+1})_{\mathbf{B}_i} \\
\mathbf{d}_i^{(\tilde{j})} &= (\boxed{0^{N_i}},\ s_i,\ 0^{N_i},\ \mathbf{y}_i^{(1,\tilde{j})},\ \mu,\ \pi_i^{(\tilde{j})},\ 0,\ 0^{2N_i+1})_{\mathbf{B}_i^*}
\end{aligned}$$

and we have $|\Pr[\mathsf{G}_6 = 1] - \Pr[\mathsf{G}_5 = 1]| \leq 4N_i \cdot \mathbf{Adv}_{\mathbb{G}_1,\mathbb{G}_2}^{\mathsf{SXDH}}(1^\lambda)$.

The game G_6 does not depend on the challenge bit b anymore and $\Pr[\mathsf{G}_6 = 1] = 1/2$. The difference in advantages is

$$\begin{aligned}\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n},\mathcal{A}}(1^\lambda) &= |\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}| \\ &= |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_6 = 1]| \\ &\leq \sum_{i=1}^{6} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i-1} = 1]| \\ &\leq \Big(\big((q_e+1)J + (q_k+1)\tilde{J}\big)\cdot(2N_i+8) + q_k + q_e + 8N_i + 2\Big)\cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)\end{aligned}$$

and the proof is completed. □

B.6 Details about our FH-DDFE in Section 6

Security We prove the security of our FH-DDFE below.

Theorem 15. *If $\mathcal{N}$ is IND-secure, $\{F_K\}_{K\in\mathcal{K}}$ and $\{F'_{K'}\}_{K'\in\mathcal{K}'}$ are families of pseudorandom functions and the* SXDH *assumption holds in $(\mathbb{G}_1,\mathbb{G}_2)$, then $\mathcal{E}$ is adaptively one-challenge weakly function-hiding secure under static corruption against complete queries in the ROM.*
More precisely, let q_h be the maximum number of queries to the oracle $\mathcal{O}$HonestGen and let q_u be an upper bound on the number of distinct sets $\mathcal{U} \subseteq \mathsf{ID}$ that occur in an encryption or key-generation query. Then, for any ppt *adversary $\mathcal{A}$, there exist* ppt *algorithms $\mathcal{B}_1,\ldots,\mathcal{B}_4$ such that*

$$\begin{aligned}\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda) \leq\ & q_h\cdot\mathbf{Adv}^{\mathsf{prf}}_{\{F_K\},\mathcal{B}_1}(1^\lambda) + q_h^2\cdot\mathbf{Adv}^{\mathsf{nike}}_{\mathcal{N},\mathcal{B}_2}(1^\lambda) \\ &+ q_h^2\cdot\mathbf{Adv}^{\mathsf{prf}}_{\{F'_{K'}\},\mathcal{B}_3}(1^\lambda) + q_u\cdot\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E}',\mathcal{F}^{\mathsf{ip}},\mathcal{B}_4}(1^\lambda)\end{aligned}$$

Proof. The proof is done via a sequence of hybrid games.

Game G_0: This is the game $\mathbf{Exp}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

Game G_1: In all key-generation and encryption queries of the form $(i,*,*)$ with $i\in\mathcal{H}$, we replace F_{K_i} with a random function R_i. By the security of the PRF, we have $|\Pr[\mathsf{G}_1=1]-\Pr[\mathsf{G}_0=1]| \leq q_h\cdot\mathbf{Adv}^{\mathsf{prf}}_{\{F_K\},\mathcal{B}_1}(1^\lambda)$.

Game G_2: For all $i,j\in\mathcal{H}$, we choose $K'_{i,j}=K'_{j,i}\xleftarrow{\$}\mathcal{K}'$ instead of $K'_{i,j}\leftarrow\mathcal{N}.\mathsf{SharedKey}(\mathcal{N}.\mathsf{sk}_i,\mathcal{N}.\mathsf{pk}_j)$ when replying to key-generation and encryption queries. The indistinguishability directly follows from the security of the NIKE scheme. Specifically, we have $|\Pr[\mathsf{G}_2=1]-\Pr[\mathsf{G}_1=1]| \leq q_h^2\cdot\mathbf{Adv}^{\mathsf{nike}}_{\mathcal{N},\mathcal{B}_2}(1^\lambda)$.

Game G_3: For all $i,j\in\mathcal{H}$, we replace $F'_{K'_{i,j}}=F'_{K'_{j,i}}$ with a random function $R'_{i,j}=R'_{j,i}$ in all key-generation and encryption queries. The indistinguishability directly follows from the security of the PRF. More precisely, we have that $|\Pr[\mathsf{G}_2=1]-\Pr[\mathsf{G}_1=1]| \leq q_h^2\cdot\mathbf{Adv}^{\mathsf{prf}}_{\{F'_{K'}\},\mathcal{B}_3}(1^\lambda)$.

Note that in G_3, for each set $\mathcal{U}$ that occurs in an encryption or key-generation query, $(\tilde{s}_i)_{i\in\mathcal{U}\cap\mathcal{H}}$ and $(\tilde{t}_i)_{i\in\mathcal{U}\cap\mathcal{H}}$ are uniformly random subject to the condition that $\sum_{i\in\mathcal{U}\cap\mathcal{H}}\tilde{s}_i = -\sum_{i\in\mathcal{U}\cap\mathcal{C}}\tilde{s}_i$ and $\sum_{i\in\mathcal{U}\cap\mathcal{H}}\tilde{t}_i = -\sum_{i\in\mathcal{U}\cap\mathcal{C}}\tilde{t}_i$.
Let $\mathcal{U}_1,\ldots,\mathcal{U}_{q_u}$ denote the q_u different sets that occur in an encryption or key-generation query where they are sorted e.g. ascending in the order in which they are queried for the first time. For $\kappa\in[0;q_u]$, we define the hybrids $\hat{\mathsf{G}}_\kappa$ as follows:

Game $\hat{\mathsf{G}}_\kappa$ for $\kappa \in [0; q_u]$: This game is the same as G_3 except that the generation of ciphertexts and secret keys is modified as follows. Upon receiving a query $\mathcal{O}\mathsf{KeyGen}(i, (\mathbf{y}_i^0, (\mathcal{U}_j, \mathsf{tag\text{-}f})), (\mathbf{y}_i^1, (\mathcal{U}_j, \mathsf{tag\text{-}f})))$, the simulator computes

$$\mathbf{d}_i \leftarrow \begin{cases} \mathcal{E}'.\mathsf{DKeyGen}(\mathsf{ek}_i, \mathsf{tag\text{-}f}, y_i^0) & \text{if } j \le \kappa \\ \mathcal{E}'.\mathsf{DKeyGen}(\mathsf{ek}_i, \mathsf{tag\text{-}f}, y_i^b) & \text{if } j > \kappa \end{cases} .$$

Similarly, upon receiving a query of the form $\mathcal{O}\mathsf{Enc}(i, (\mathbf{x}_i^0, (\mathcal{U}_j, \mathsf{tag})), (\mathbf{x}_i^1, (\mathcal{U}_j, \mathsf{tag})))$, the simulator computes

$$\mathbf{c}_i \leftarrow \begin{cases} \mathcal{E}'.\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i^0) & \text{if } j \le \kappa \\ \mathcal{E}'.\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i^b) & \text{if } j > \kappa \end{cases} .$$

Note that $\hat{\mathsf{G}}_0 = \mathsf{G}_3$ and $\hat{\mathsf{G}}_{q_u}$ is independent of the bit b. For $\kappa \in [q_u]$, we have $|\Pr[\hat{\mathsf{G}}_\kappa = 1] - \Pr[\hat{\mathsf{G}}_{\kappa-1} = 1]| \le \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E}', \mathcal{F}^{\mathsf{ip}}, \mathcal{B}_4}(1^\lambda)$ by Lemma 14. □

B.7 From Complete to Incomplete Challenges - Proof of Lemma 16

Lemma 16. *Assume there exist (1) a one-challenge, weakly function-hiding DDFE scheme $\mathcal{E}^{\mathsf{pos}}$ for the function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that is secure against complete queries, and (2) an AoNE scheme $\mathcal{E}^{\mathsf{aone}}$ whose message space contains the ciphertext space of $\mathcal{E}^{\mathsf{pos}}$. Then there exists a one-challenge, weakly function-hiding DDFE scheme $\mathcal{E}$ for the same function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that is even secure against incomplete queries.*
More precisely, for any ppt *adversary $\mathcal{A}$, there exist* ppt *algorithms $\mathcal{B}$ and $\mathcal{B}'$ such that*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E}, \mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}, \mathcal{A}}(1^\lambda) \le 6 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}xxx\text{-}wfh}}_{\mathcal{E}^{\mathsf{pos}}, \mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}, \mathcal{B}}(1^\lambda) + 6 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}^{\mathsf{aone}}, \mathcal{F}^{\mathsf{aone}}, \mathcal{B}'}(1^\lambda) ,$$

where $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}\}$.

Proof. Let $\mathcal{E}^{\mathsf{pos}} = (\mathsf{pGSetup}, \mathsf{pLSetup}, \mathsf{pKeyGen}, \mathsf{pEnc}, \mathsf{pDec})$ be a one-challenge, weakly function-hiding DDFE scheme for the function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that is secure against complete queries, and let $\mathcal{E}^{\mathsf{aone}} = (\mathsf{aGSetup}, \mathsf{aLSetup}, \mathsf{aEnc}, \mathsf{aDec})$ be a DDFE scheme for the AoNE functionality $\mathcal{F}^{\mathsf{aone}}$. We construct a one-challenge, weakly function-hiding DDFE scheme $\mathcal{E}$ for the function class $\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}}$ that is secure against incomplete queries. The details of $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ go as follows:

$\mathsf{GSetup}(1^\lambda)$**:** On input the security parameter 1^λ, run

$$\mathsf{pPP} \leftarrow \mathsf{pGSetup}(1^\lambda); \quad \mathsf{aPP} \leftarrow \mathsf{aGSetup}(1^\lambda)$$

and return $\mathsf{PP} := (\mathsf{pPP}, \mathsf{aPP})$

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

$$(\mathsf{pSK}_i, \mathsf{pPK}_i) \leftarrow \mathsf{pLSetup}(\mathsf{pPP}); \quad (\mathsf{aSK}_i, \mathsf{aPK}_i) \leftarrow \mathsf{aLSetup}(\mathsf{aPP})$$

and return $(\mathsf{SK}_i := (\mathsf{pSK}_i, \mathsf{aSK}_i), \mathsf{PK}_i := (\mathsf{pPK}_i, \mathsf{aPK}_i))$.

$\mathsf{KeyGen}(\mathsf{SK}_i, k)$**:** On input a secret key SK_i and $k = (k_{\mathrm{pri}}, k_{\mathrm{pub}})$, compute

$$\mathsf{pDK}_i \leftarrow \mathsf{pKeyGen}(\mathsf{pSK}_i, k); \quad \mathsf{aDK}_i \leftarrow \mathsf{aEnc}(\mathsf{aSK}_i, (i, (\mathsf{pDK}_i, k_{\mathrm{pub}})))$$

and return $\mathsf{DK}_i := \mathsf{aDK}_i$.

$\mathsf{Enc}(\mathsf{SK}_i, m)$**:** On input a secret key SK_i and $m = (m_{\mathrm{pri}}, m_{\mathrm{pub}})$, compute:

$$\mathsf{pCT}_i \leftarrow \mathsf{pEnc}(\mathsf{pSK}_i, m); \quad \mathsf{aCT}_i \leftarrow \mathsf{aEnc}(\mathsf{aSK}_i, (\mathsf{pCT}_i, m_{\mathrm{pub}}))$$

and return $\mathsf{CT}_i \coloneqq \mathsf{aCT}_i$.

$\mathsf{Dec}((\mathsf{DK}_i)_{i\in\mathcal{U}_K}, (\mathsf{CT}_i)_{i\in\mathcal{U}_M})$**:** On input a set of secret keys $(\mathsf{DK}_i)_{i\in\mathcal{U}_K}$ and a set of ciphertexts $(\mathsf{CT}_i)_{i\in\mathcal{U}_M}$, compute

$$(\mathsf{pDK}_i)_{i\in\mathcal{U}_K} \leftarrow \mathsf{aDec}((\mathsf{aDK}_i)_{i\in\mathcal{U}}); \quad (\mathsf{pCT}_i)_{i\in\mathcal{U}_M} \leftarrow \mathsf{aDec}((\mathsf{aCT}_i)_{i\in\mathcal{U}}) \ .$$

If one of these decryption processes returns $\perp$, return the same value. Otherwise, return $\mathsf{out} \leftarrow \mathsf{pDec}(\{\mathsf{pDK}_i\}_{i\in\mathcal{U}_K}, \{\mathsf{pCT}_i\}_{i\in\mathcal{U}_M})$.

The correctness of $\mathcal{E}$ follows immediately from the correctness of $\mathcal{E}^{\mathsf{pos}}$ and $\mathcal{E}^{\mathsf{aone}}$. Turning to its security, we introduce a sequence of hybrids $\mathsf{G}_0, \ldots, \mathsf{G}_4$. For $i \in [0;4]$, we denote $\mathbf{Adv}^{\mathsf{G}_i}(\mathcal{A}) \coloneqq |\Pr[\mathsf{G}_i = 1] - 1/2|$. To improve readability, we introduce the shorthands

$$\left(\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}, \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}}\right) \coloneqq \left(\mathbf{Exp}^{\mathsf{pos\text{-}xxx\text{-}wfh}}_{\mathcal{E}^{\mathsf{pos}},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{B}}(1^\lambda), \mathbf{Adv}^{\mathsf{pos\text{-}xxx\text{-}wfh}}_{\mathcal{E}^{\mathsf{pos}},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{B}}(1^\lambda)\right)$$
$$(\mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}'}, \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}'}) \coloneqq \left(\mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}^{\mathsf{aone}},\mathcal{F}^{\mathsf{aone}},\mathcal{B}'}(1^\lambda), \mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}^{\mathsf{aone}},\mathcal{F}^{\mathsf{aone}},\mathcal{B}'}(1^\lambda)\right) \ .$$

The hybrid games are defined as follows.

Game G_0**:** This game equals $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{F},\mathcal{F},\mathcal{A}}(1^\lambda)$, i.e. $\mathbf{Adv}^{\mathsf{G}_0} = \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

Recall that one-challenge security states that the adversary must declare up front to $\mathsf{Initialize}$ additional public information for challenge messages and challenge keys $m^*_{\mathrm{pub}} = (\mathcal{U}^*_M, \mathsf{tag}^*), k^*_{\mathrm{pub}} = (\mathcal{U}^*_K, \mathsf{tag\text{-}f}^*)$ so that:

- if $(i, m^{(0)}_i, m^{(1)}_i) \in \mathcal{Q}_{\mathsf{Enc}}$ and $m^{(0)}_{i,\mathrm{pub}} = m^{(1)}_{i,\mathrm{pub}} \neq m^*_{\mathrm{pub}}$, then $m^{(0)}_i = m^{(1)}_i$,
- if $(i, k^{(0)}_i, k^{(1)}_i) \in \mathcal{Q}_{\mathsf{KGen}}$ and $k^{(0)}_{i,\mathrm{pub}} = k^{(1)}_{i,\mathrm{pub}} \neq k^*_{\mathrm{pub}}$, then $k^{(0)}_i = k^{(1)}_i$.

We define events E_0 and E_1 as follows:

(E_0) $\mathcal{A}$ has asked queries of the form $\mathcal{O}\mathsf{KeyGen}(i, (*, k^*_{\mathrm{pub}}), (*, k^*_{\mathrm{pub}}))$ for all or no $i \in \mathcal{H} \cap \mathcal{U}^*_K$.
(E_1) $\mathcal{A}$ has asked queries of the form $\mathcal{O}\mathsf{KeyGen}(i, (*, k^*_{\mathrm{pub}}), (*, k^*_{\mathrm{pub}}))$ for some but not all $i \in \mathcal{H} \cap \mathcal{U}^*_K$, i.e. $E_1 = \neg E_0$.

Game G_1**:** This is the same as G_0 except that the simulator chooses a random bit $d \xleftarrow{\$} \{0,1\}$ during $\mathsf{Initialize}$. Upon $\mathcal{A}$ calling $\mathsf{Finalize}$, if ($d = 0$ and E_1 happens) or ($d = 1$ and E_0 happens), the simulator outputs a random bit and aborts. Note that the simulator's behavior is independent of the bit d before $\mathsf{Finalize}$ is called. Therefore, we have $\mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A}) = 1/2 \cdot \mathbf{Adv}^{\mathsf{G}_0}(\mathcal{A})$.

Game G_2**:** If $d = 1$, then the simulation works exactly as in the previous game. Otherwise, the simulator acts as an adversary $\mathcal{B}$ in the game $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}$. Specifically, if $d = 0$, the simulation works as follows:

- *Initialization:* Upon $\mathcal{A}$ calling $\mathsf{Initialize}(1^\lambda, m^*_{\mathrm{pub}}, k^*_{\mathrm{pub}})$, $\mathcal{B}$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$, runs

$$\mathsf{pPP} \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathsf{Initialize}(1^\lambda); \quad \mathsf{aPP} \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{GSetup}(1^\lambda)$$

and returns $\mathsf{PP} \coloneqq (\mathsf{pPP}, \mathsf{aPP})$.

- *User-Generation Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{HonestGen}(i)$ for some $i \in \mathsf{ID}$, $\mathcal{B}$ queries and computes

$$\mathsf{pPK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{HonestGen}(i); \quad (\mathsf{aSK}_i, \mathsf{aPK}_i) \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{LSetup}(i) \ ,$$

  adds i to $\mathcal{H}$ and returns $\mathsf{PK}_i := (\mathsf{pPK}_i, \mathsf{aPK}_i)$.
- *Corruption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Corrupt}(i)$ for some $i \in \mathsf{ID}$, if $i \notin \mathcal{H}$, then $\mathcal{B}$ first calls $\mathcal{O}\mathsf{HonestGen}(i)$. Then it queries $\mathsf{pSK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{Corrupt}(i)$ and returns $\mathsf{SK}_i := (\mathsf{pSK}_i, \mathsf{aSK}_i)$ where aSK_i is known from the corresponding query to $\mathcal{O}\mathsf{HonestGen}$.
- *Encryption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i, (m^0_{\mathrm{pri}}, m_{\mathrm{pub}}), (m^1_{\mathrm{pri}}, m_{\mathrm{pub}}))$, $\mathcal{B}$ queries and computes

$$\mathsf{pCT}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{Enc}(i, (m^b_{\mathrm{pri}}, m_{\mathrm{pub}}), (m^b_{\mathrm{pri}}, m_{\mathrm{pub}})); \quad \mathsf{aCT}_i \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{Enc}(\mathsf{aSK}_i, (i, (\mathsf{pCT}_i, m_{\mathrm{pub}})))$$

  and returns $\mathsf{CT}_i := \mathsf{aCT}_i$.
- *Key-Generation Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}$ on input $(i, (k^0_{\mathrm{pri}}, k_{\mathrm{pub}}), (k^1_{\mathrm{pri}}, k_{\mathrm{pub}}))$, $\mathcal{B}$ does the following:
  - If $k_{\mathrm{pub}} = k^*_{\mathrm{pub}}$, $\mathcal{B}$ queries

$$\mathsf{pDK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{KeyGen}(i, (k^b_{\mathrm{pri}}, k^*_{\mathrm{pub}}), (k^1_{\mathrm{pri}}, k^*_{\mathrm{pub}})); \quad \mathsf{aDK}_i \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{Enc}(\mathsf{aSK}_i, (i, (\mathsf{pDK}_i, k^*_{\mathrm{pub}})))$$

    and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.
  - If $k_{\mathrm{pub}} \neq k^*_{\mathrm{pub}}$, then $k^0_{\mathrm{pri}} = k^1_{\mathrm{pri}}$ and $\mathcal{B}$ queries

$$\mathsf{pDK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{KeyGen}(i, (k^1_{\mathrm{pri}}, k^*_{\mathrm{pub}}), (k^1_{\mathrm{pri}}, k^*_{\mathrm{pub}})); \quad \mathsf{aDK}_i \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{Enc}(\mathsf{aSK}_i, (i, (\mathsf{pDK}_i, k^*_{\mathrm{pub}})))$$

    and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.
- *Finalize:* Upon $\mathcal{A}$ calling $\mathsf{Finalize}(b')$, $\mathcal{B}$ forwards the same bit to its own challenger by calling $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathsf{Finalize}(b')$.

In the end, we have $|\mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A})| \leq \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}}$.

Game G_3: We do a similar modification in the simulation for the case $d = 1$. That is, for queries of the form $\mathcal{O}\mathsf{KeyGen}((i, (k^0_{\mathrm{pri}}, k^*_{\mathrm{pub}}), (k^1_{\mathrm{pri}}, k^*_{\mathrm{pub}})))$, the simulator now outputs a key for $(k^1_{\mathrm{pri}}, k^*_{\mathrm{pub}})$ instead of $(k^b_{\mathrm{pri}}, k^*_{\mathrm{pub}})$. The indistinguishability between G_3 and G_2 reduces to the security of $\mathcal{E}^{\mathsf{aone}}$. We construct a reduction $\mathcal{B}'$ that acts as an adversary in the experiment $\mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}'}$. $\mathcal{B}'$ replaces all $\mathcal{E}^{\mathsf{aone}}$ algorithms with calls to the respective oracles of $\mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}'}$, in a similar way as $\mathcal{B}$ does with calls to oracles of $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}$ in G_2. In particular, upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}$ on input $(i, (k^0_{\mathrm{pri}}, k_{\mathrm{pub}}), (k^1_{\mathrm{pri}}, k_{\mathrm{pub}}))$, $\mathcal{B}$ does the following:

- If $k_{\mathrm{pub}} = k^*_{\mathrm{pub}}$, $\mathcal{B}$ queries

$$\begin{aligned}
\mathsf{pDK}^b_i &\leftarrow \mathcal{E}^{\mathsf{pos}}.\mathsf{KeyGen}(\mathsf{pSK}_i, (k^b_{\mathrm{pri}}, k^*_{\mathrm{pub}})) \\
\mathsf{pDK}^1_i &\leftarrow \mathcal{E}^{\mathsf{pos}}.\mathsf{KeyGen}(\mathsf{pSK}_i, (k^1_{\mathrm{pri}}, k^*_{\mathrm{pub}})) \\
\mathsf{aDK}_i &\leftarrow \mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}'}.\mathcal{O}\mathsf{Enc}(i, (\mathsf{pDK}^b_i, k^*_{\mathrm{pub}}), (\mathsf{pDK}^1_i, k^*_{\mathrm{pub}}))
\end{aligned}$$

  and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.
- If $k_{\mathrm{pub}} \neq k^*_{\mathrm{pub}}$, then $k^0_{\mathrm{pri}} = k^1_{\mathrm{pri}}$ and $\mathcal{B}$ queries

$$\begin{aligned}
\mathsf{pDK}_i &\leftarrow \mathcal{E}^{\mathsf{pos}}.\mathsf{KeyGen}(\mathsf{pSK}_i, (k^1_{\mathrm{pri}}, k_{\mathrm{pub}})) \\
\mathsf{aDK}_i &\leftarrow \mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}'}.\mathcal{O}\mathsf{Enc}(i, (\mathsf{pDK}_i, k_{\mathrm{pub}}), (\mathsf{pDK}_i, k_{\mathrm{pub}}))
\end{aligned}$$

  and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.

In the end, we have $|\mathbf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A})| \leq \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}'}$.

Game G_4: We answer queries of the form $\mathcal{O}\mathsf{Enc}((i,(m^0_{\mathrm{pri}},m^*_{\mathrm{pub}}),(m^1_{\mathrm{pri}},m^*_{\mathrm{pub}})))$ by encryptions of $(m^1_{\mathrm{pri}},m^*_{\mathrm{pub}})$ as opposed to $(m^b_{\mathrm{pri}},m^*_{\mathrm{pub}})$ using a similar sequence of hybrids as G_1, G_2 and G_3, but with flipped roles of the oracles $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Enc}$. Note that G_4 is independent of the bit b. In the end, we obtain $\mathbf{Adv}^{\mathsf{G}_3}(\mathcal{A}) \leq 2\cdot(\mathbf{Adv}^{\mathsf{G}_4}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}} + \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}'})$

To conclude, we compute

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{\mathsf{dyn}},\mathcal{A}}(1^\lambda) &= \mathbf{Adv}^{\mathsf{G}_0}(\mathcal{A}) \\
&= 2\cdot\mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A}) \\
&\leq 2\cdot(\mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}}) \\
&\leq 2\cdot(\mathbf{Adv}^{\mathsf{G}_3}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}} + \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}'}) \\
&\leq 4\cdot\mathbf{Adv}^{\mathsf{G}_4}(\mathcal{A}) + 6\cdot\mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}} + 6\cdot\mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}'} \\
&= 6\cdot\mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}} + 6\cdot\mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}'}\ ,
\end{aligned}$$

where the last equality follows from the fact that G_4 is independent of b. □