# Dynamic Decentralized Functional Encryption with Strong Security

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 first provide a generic conversion from DMCFE to DDFE, that preserves the security properties, in both the standard and the function-hiding setting. Then, new proof techniques allow us to analyze a new concrete construction of function-hiding DMCFE for inner products, that can thereafter be converted into a DDFE, 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.

## Table of Contents

## 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 [55, 22], 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 [56, 30, 21] or Attribute-Based Encryption [55, 40, 54, 14, 53], and has become a very active domain of research. Abdalla *et al.* [3] proposed the first FE scheme (ABDP scheme) that allows computing the inner product between a functional vector in the functional decryption key and a data vector in the ciphertext, thenceforth coined $\mathsf{IPFE}$. The interests in FE then increased, either in improving existing constructions for concrete function classes, *e.g.* inner products [10, 17, 24] and quadratic functions [15, 36, 13, 46], or in pushing the studies of new advanced notions [38] as well as the relationship to other notions in cryptography [12, 19]. 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.* [37, 39] initiated the study of *Multi-Input Functional Encryption* ($\mathsf{MIFE}$) and *Multi-Client Functional Encryption* ($\mathsf{MCFE}$). In $\mathsf{MCFE}$ 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* $\mathsf{ek}_i$ to encrypt $x_i$, under some (usually time-based) tag $\mathsf{tag}$. 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 $\mathsf{dk}_F$ 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 $\mathsf{tag}$. No information beyond $F(x_1, \ldots, x_n)$ is leaked, especially concerning the individual secret components $x_i$, 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 $\mathsf{ek}_i$ being private. The notion of $\mathsf{MCFE}$ 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, $\mathsf{MIFE}/\mathsf{MCFE}$ motivated a plethora of works on the subject, notably for the concrete function class of inner products [33, 27, 28, 4, 2, 1, 45, 29, 5, 49].

*Decentralized Multi-Client Functional Encryption.* The setup of $\mathsf{MCFE}$ requires some authority (a trusted third party) responsible for the setup and generation of functional decryption keys. The authority possesses a master secret key $\mathsf{msk}$ that can be used to handle the distribution of private encryption keys $\mathsf{ek}_i$ and deriving functional decryption keys $\mathsf{dk}_F$. 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.* [27]. In $\mathsf{DMCFE}$, 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* $\mathsf{sk}_i$. Agreeing on a function $F$,

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

each sender generates their functional key $\mathsf{dk}_{F,i}$ using $\mathsf{sk}_i$, the description of $F$, and a tag $\mathsf{tag\text{-}f}$. Originally in [27], 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 [27] constructed the first DMCFE for computing inner products (IP-DMCFE), 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 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 [27] on both aspects of efficiency as well as security, or by giving generic transformation to (D)MCFE from single-client FE [45, 2, 1].

*Dynamic Decentralized Functional Encryption.* In [29], Chotard *et al.* generalized DMCFE and defined the notion of *Dynamic Decentralized Functional Encryption* (DDFE) that allows users[2] 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 is *a priori* interactive. When joining a DDFE system, each user $i$ 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 $\mathsf{sk}_i$. A set $\mathcal{U}_M$ of users can use $\mathsf{sk}_i$ to independently encrypt their data, contributing to a list of ciphertexts $(\mathsf{ct}_i)_{i\in\mathcal{U}_M}$. In the same way, a set $\mathcal{U}_K$ of users can use their $\mathsf{sk}_i$ to independently contribute to a list of 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, while requiring no trusted third party. Being dynamic, a DDFE scheme does not demand in advance a fixed number of users in $\mathcal{U}_M$ nor $\mathcal{U}_K$. The authors of [29] provide a concrete construction of DDFE for the function class computing inner products (IP-DDFE). A recent work by Agrawal *et al.* [8] revisits the notion of DDFE and provides a construction of IP-DDFE with stronger security guarantees (see below). To date, these works provide the only two known IP-DDFE candidates in the literature.

*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* [6] and *Multi-Authority Attribute-Based Encryption* [25]. Interestingly, Agrawal *et al.* [8] 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.

**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 a 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 *function-hiding* property.

[2] The terminology for the participants in the various FE models is not unique in the literature. The encryptors $i$ in MCFE are usually referred to as *clients* who hold an *encryption key* $\mathsf{ek}_i$. This term is adopted in [27] for DMCFE, but the key-generating parties are referred to as *senders* who possess *secret keys* $\mathsf{sk}_i$. In the definition of DDFE introduced in [29], encryption and key generation are performed by the same parties, which are then simply referred to as *users*, and which have *secret keys* $\mathsf{sk}_i$. We follow these namings accordingly when discussing DDFE and DMCFE.

Besides practical applications, function-hiding FE schemes for restricted function classes (such as inner products) have also proven to be an important technical building block for the construction of FE schemes for broader function classes: Lin [46] employed a function-hiding IPFE (FH-IPFE) to obtain an FE scheme for quadratic functions. A different technique was also introduced by Gay in [36] equally aiming at constructing FE for quadratic functions. With several technical novelties, Agrawal *et al.* [7, 9] were able to generalize the aforementioned constructions to obtain MIFE for quadratic functions.

*Existing Function-Hiding FE Schemes in the Literature.* Bishop *et al.* [18] presented the first IPFE 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.* [31, 32]. This was further improved in terms of efficiency and/or computational hardness assumptions by works of [59, 42, 41]. The constructions of [18, 31, 59] all leverage the power of *dual pairing vector spaces* (DPVSes) developed by Okamoto and Takashima in [51, 52, 53]. In 2017, Lin [46] used a different approach that used the ABDP IPFE to get simpler constructions of FH-IPFE. Using the same blueprint and exploiting the specific algebraic properties of the underlying inner-product MIFE scheme carefully, Abdalla *et al.* [4] were able to construct function-hiding MIFE for inner products (FH-IP-MIFE). In [8], Agrawal *et al.* came up with the first construction of function-hiding MCFE for inner products (FH-IP-MCFE) that is inspired by the FH-IP-MIFE by Datta *et al.* [33]. Very recently, Shi and Vanjani [57] presented a generic transformation from single-client to multi-client functional encryption, preserving the function-hiding property and leading to the first FH-IP-MCFE with adaptive security. Remarkably, their security proof does not rely on random oracles. We are not aware of any construction of function-hiding DMCFE or DDFE for inner products (FH-IP-DMCFE or FH-IP-DDFE) whose security proof does not use the *random oracle model* (ROM). Again in [8], the authors were able to lift their aforementioned FH-IP-MCFE scheme to FH-IP-DDFE, following the approach of Chotard *et al.* [29] who presented a similar transformation in the non-function-hiding setting. So far, the scheme of [8] is the only FH-IP-DDFE in the literature. They achieve indistinguishability-based security in the ROM.

All known constructions that guarantee function-hiding security rely on pairings. A recent work by Ünal [60] shows that in the manner of most lattice-based approaches, there is little hope to attain function privacy in IPFE schemes, in the setting of multi-user or not.

**Repetitions under One Tag.** Involving tags at the time of encryption and key-generation restricts that only ciphertexts and 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 [27] and subsequent works [45], once the adversary makes a query for $(i, \mathsf{tag})$, further queries for the same pair $(i, \mathsf{tag})$ will be ignored. This means repetitions are not taken into account. The authors of [27] argued that it is the responsibility of the users not to use the same tag twice. However, a security notion for 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 [2, 1]. No tags exist in the general definition of DDFE. However, jumping ahead, the specific inner product functionality considered in this paper involves tags, so the previous reasoning in the context of DMCFE carries over to the DDFE case as well. In addition, when repetitions are allowed for ciphertexts, the security model of MCFE encompasses MIFE, where there is no tag, by just replacing tags with a constant value.

## 1.1 Our Contributions

To the best of our knowledge, the only candidates of DDFE for the class of inner products come from [29] (standard security) and [8] (function-hiding security). Both constructions achieve only *selective* security under *static* corruption, *i.e.* the adversary makes all encryption, key-generation and corruption queries up front in one shot. [29] allows repetitions for both encryption and key-generation queries, whereas [8] allows them only for encryption. This state-of-the-art leads us to the following question:

*How far can we raise the security level of (function-hiding)* DDFE?

In this paper, we give several candidates for IP-DDFE that strictly improve on various aspects of security compared with [29, 8]. Below and in Table 1 are presented a summary of our contributions and a comparison with existing works:

1. *From* DMCFE *to* DDFE. Inspired by the (non-generic) lifting result from FH-IP-MCFE to FH-IP-DDFE of [8], we present a generic conversion from DMCFE to DDFE that works both in the standard and function-hiding setting. Our conversion mostly preserves the security level of the underlying DMCFE scheme (*e.g.* adaptive oracle queries and repetitions) or even improves it: we are able to generically remove the so-called complete-query constraint, a restriction that many previous (D)MCFE schemes suffered from [27, 45] (see constraint 4 of Definition 4). The only restriction in the security model that we do not know how to avoid in our constructions is static corruption.
   By demonstrating how to plug various IP-DMCFE schemes into our generic conversion, we obtain a number of new IP-DDFE candidates with previously unattained properties that are *emphasized* in what follows. Specifically, we employ an FH-IP-DMCFE scheme that is also constructed in this work (see item 2 below) to obtain an FH-IP-DDFE where both encryption and key-generation queries can be made *adaptively* and *with repetitions*. The security is based on SXDH in the ROM. Furthermore, we show that the IP-DMCFE schemes of [27, 45] can be lifted to IP-DDFE using our conversion. The latter gives us the first *adaptively secure* IP-DDFE scheme *based on* LWE in the *standard model*. Since the IP-DMCFE of [45] does not support repetitions, the obtained IP-DDFE does neither. A high-level overview and the technical details are given in Sections 3 and 4, respectively.
2. *Function-Hiding* IP-DMCFE. As a stepping stone to our FH-IP-DDFE, we construct the first FH-IP-DMCFE that tolerates *adaptive* encryption queries (with repetitions) and *adaptive* key-generation queries *with repetitions*, under static corruption. It uses pairings and is provably secure in the ROM. The only existing FH-IP-DMCFE in the literature is the one that follows implicitly from the FH-IP-DDFE of [8] and thus inherits the restrictions of the security model, *i.e.* selective oracle queries, no repetitions for key-generation queries and static corruption. The high-level ideas of our construction are explained in Section 3.1, details can be found in Section C.1.
3. *Technical Contribution.* Along the way, we push forward the study of DPVS techniques. We state a novel lemma that shows the indistinguishability of two distributions in a setting where not all input data is known up front. This lemma proves to be the key ingredient for the security proof of our FH-IP-DMCFE scheme in the adaptive setting. Due to its generality, we believe that the lemma can find other applications in the future. The formal statement can be found in Lemma 34 in Section C.2. An overview on how the lemma is used is included in Section 3.1. Basic definitions for the DPVS framework are provided in Section 2.1

## 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 $[n] := [1; n]$. For a finite set $S$, we let $2^{\mathcal{S}}$ denote the power set of $\mathcal{S}$, and $U(S)$ denote the

| Scheme | Type | FH | Oracle Queries | | Assumptions | ROM |
|---|---|---|---|---|---|---|
| | | | $\mathcal{O}$Enc | $\mathcal{O}$KeyGen | | |
| [8, Section 6.2] | IP-MCFE | $\underline{\checkmark}$ | sel, $\underline{\text{w-rep}}$ | sel$^\dagger$ | SXDH, pairings | ✓ |
| [57, Section B.3] | IP-MCFE | $\underline{\checkmark}$ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | $\underline{\text{adap}}^\dagger$ | D-Lin, pairings | $\underline{\text{✗}}^\ddagger$ |
| Section C.1 | IP-DMCFE | $\underline{\checkmark}$ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | SXDH, pairings | ✓ |
| [29, Section 7] | IP-DDFE | ✗ | sel, $\underline{\text{w-rep}}$ | sel, $\underline{\text{w-rep}}$ | DDH | ✓ |
| Section 4 + B.4 | IP-DDFE | ✗ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | SXDH, pairings | ✓ |
| Section 4 + B.5 | IP-DDFE | ✗ | $\underline{\text{adap}}$, no-rep | $\underline{\text{adap}}$, no-rep | LWE | $\underline{\text{✗}}$ |
| [8, Section 6.3] | IP-DDFE | $\underline{\checkmark}$ | sel, $\underline{\text{w-rep}}$ | sel, no-rep | SXDH, pairings | ✓ |
| Section 4 + B.3 | IP-DDFE | $\underline{\checkmark}$ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | $\underline{\text{adap}}$, $\underline{\text{w-rep}}$ | SXDH, pairings | ✓ |

$^\dagger$ For MCFE, there is no notion of tags for key generation, hence no notion of repetitions.
$^\ddagger$ This is the only FH-MCFE that is provably secure without the ROM. To our knowledge, there is no FH-DMCFE nor FH-DDFE in the literature that does not use ROs. In Remark 10, we discuss whether our constructions could be made non-ROM using similar techniques.

Table 1: We compare our constructions with existing works, in terms of the type of primitives (**Type**), whether the scheme provides standard security (✗) or the stronger function-hiding security (✓) (**FH**), whether the encryption oracle ($\mathcal{O}$Enc) and key-generation oracle ($\mathcal{O}$KeyGen) can be queried adaptively and with repetitions (**Oracle Queries**), which assumptions are used for the security proof (**Assumptions**), and whether the security is proven in the ROM (✓) or not (✗) (**ROM**). The shorthands (sel, adap) denote selective or adaptive oracle queries. The shorthands (w-rep, no-rep) indicates whether the adversary can demand repetitive queries to the same slot and tag or not. All schemes are defined for the inner-product functionality of their respective type of primitive (see Def. 2 and 8) and consider only static corruption. Preferred properties are $\underline{\text{underlined}}$.

uniform distribution over $S$. 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 $\mathbb{Z}_q$, and use non-boldface capital letters $B, H, \ldots$ for scalar matrices in $GL_N(\mathbb{Z}_q)$. 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 [34] 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 boldface letters $\mathbf{B}, \mathbf{b}, \ldots$ for matrices and vectors of group elements. We use the shorthand ppt for "probabilistic polynomial time". In the security proofs, whenever we use an ordered sequence of games $(\mathsf{G}_0, \mathsf{G}_1, \ldots, \mathsf{G}_i, \ldots, \mathsf{G}_L)$ indexed by $i \in [0; L]$, we refer to the predecessor of $\mathsf{G}_j$ by $\mathsf{G}_{j-1}$, for $j \in [L]$. We recall the hardness assumption used throughout this work in Appendix A.1.

## 2.1 Dual Pairing Vector Spaces

Our constructions rely on the *Dual Pairing Vector Spaces* (DPVS) framework in the 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 [51, 52, 53] aiming at adaptive security for ABE as well as IBE, together with the *dual system methodology* introduced by Waters [61]. In [44], the setting for dual systems is composite-order bilinear groups. Continuing on this line of works, Chen *et al.* [26] used prime-order bilinear groups under the SXDH assumption.

**Formalizations.** Let us fix $N \in \mathbb{N}$ and consider $\mathbb{G}_1^N$ having $N$ copies of $\mathbb{G}_1$. Viewing $\mathbb{Z}_q^N$ as a vector space of dimension $N$ over $\mathbb{Z}_q$ with the notions of bases, we can obtain naturally a similar notion of bases for $\mathbb{G}_1^N$. More specifically, any invertible matrix $B \in GL_N(\mathbb{Z}_q)$ identifies a basis $\mathbf{B}$ of $\mathbb{G}_1^N$, whose $i$-th row $\mathbf{b}_i$ is $[\![B_i]\!]_1$, where $B_i$ is the $i$-th row of $B$. It is straightforward that we can write $\mathbf{B} = [\![B]\!]_1$ for any basis $\mathbf{B}$ of $\mathbb{G}_1^N$ corresponding to an invertible matrix $B \in GL_N(\mathbb{Z}_q)$. We write $\mathbf{x} = (m_1, \ldots, m_N)_{\mathbf{B}}$ to indicate the representation of $\mathbf{x}$ in the basis $\mathbf{B}$, i.e. $\mathbf{x} = \sum_{i=1}^N m_i \cdot \mathbf{b}_i$. At some point when we focus on the indices in an ordered list $L$ of length $\ell$, 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 $\mathbf{x} \times \mathbf{y} := \prod_{i=1}^N \mathbf{e}(\mathbf{x}[i], \mathbf{y}[i]) = [\![\langle (m_1, \ldots, m_N), (k_1, \ldots, k_N)\rangle]\!]_{\mathsf{t}}$. Given a basis $\mathbf{B} = (\mathbf{b}_i)_{i \in [N]}$ of $\mathbb{G}_1^N$, we define $\mathbf{B}^*$ to be a basis of $\mathbb{G}_2^N$ by first defining $B^* := (B^{-1})^\top$ and the $i$-th row $\mathbf{b}_i^*$ of $\mathbf{B}^*$ is $[\![B_i^*]\!]_2$. It holds that $B \cdot (B^*)^\top = I_N$ the identity matrix and $\mathbf{b}_i \times \mathbf{b}_j^* = [\![\delta_{i,j}]\!]_{\mathsf{t}}$ for every $i, j \in [N]$, where $\delta_{i,j} = 1$ if and only if $i = j$. We call the pair $(\mathbf{B}, \mathbf{B}^*)$ a *pair of dual orthogonal bases* of $(\mathbb{G}_1^N, \mathbb{G}_2^N)$. If $\mathbf{B}$ is constructed by a random invertible matrix $B \xleftarrow{\$} GL_N(\mathbb{Z}_q)$, we call the resulting $(\mathbf{B}, \mathbf{B}^*)$ a pair of random dual bases. A DPVS is a bilinear group setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathsf{t}}, g_1, g_2, g_{\mathsf{t}}, \mathbf{e}, q)$ with dual orthogonal bases. We denote by $\mathsf{DPVSGen}$ 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 $\mathsf{poly}(\log q)$. Further details on DPVS-related techniques can be found in Appendix A.2.

## 2.2 Dynamic Decentralized Functional Encryption

In this section we recall the notion of *Dynamic Decentralized Functional Encryption* with standard security ($\mathsf{DDFE}$) and the strong function-hiding security ($\mathsf{FH\text{-}DDFE}$). This notion was introduced first in [29] and later defined in [8, Section 6.1] as a special case of the Multi-Party Functional Encryption notion. Let $\{\mathsf{ID}_\lambda\}_{\lambda \in \mathbb{N}}$, $\{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, $\{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{R}_\lambda\}_{\lambda \in \mathbb{N}}$ be families of identity, key, message and output spaces, respectively, where all of which are indexed by a security parameter $\lambda \in \mathbb{N}$ and $\mathcal{K}_\lambda = \mathcal{K}_{\lambda,\text{pri}} \times \mathcal{K}_{\lambda,\text{pub}}$, $\mathcal{M}_\lambda = \mathcal{M}_{\lambda,\text{pri}} \times \mathcal{M}_{\lambda,\text{pub}}$ consist of a private and a public component each. We consider a functionality

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

**Definition 1 (Dynamic Decentralized Functional Encryption).** *A* $\mathsf{DDFE}$ *scheme* $\mathcal{E}$ *for the functionality* $f^{\mathsf{dyn}}$ *comprises five algorithms:*

$\mathsf{GSetup}(1^\lambda)$**:** *Take* $1^\lambda$ *as input and output a set of public parameters* $\mathsf{pp}$.

$\mathsf{LSetup}(\mathsf{pp})$**:** *Take as input* $\mathsf{pp}$*, output a public key* $\mathsf{pk}_i$ *and a secret key* $\mathsf{sk}_i$*. We assume that* $\mathsf{pk}_i$ *implicitly contains* $\mathsf{pp}$ *and that* $\mathsf{sk}_i$ *implicitly contains* $\mathsf{pk}_i$*.*

$\mathsf{KeyGen}(\mathsf{sk}_i, k = (k_{\text{pri}}, k_{\text{pub}}))$**:** *Given a secret key* $\mathsf{sk}_i$ *and* $k \in \mathcal{K}_\lambda$*, output* $\mathsf{dk}_i$*.*

$\mathsf{Enc}(\mathsf{sk}_i, m = (m_{\text{pri}}, m_{\text{pub}}))$**:** *Given a secret key* $\mathsf{sk}_i$ *and* $m \in \mathcal{M}_\lambda$*, output* $\mathsf{ct}_i$*.*

$\mathsf{Dec}((\mathsf{dk}_i)_{i \in \mathcal{U}_K}, (\mathsf{ct}_i)_{i \in \mathcal{U}_M})$**:** *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* $\mathcal{R}_\lambda$ *or* $\perp$*.*

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

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

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

**Functionalities.** We consider DDFE for two concrete function classes.

*(Dynamic) Inner-Product Functionality.* We define the function family $f^{\mathsf{dyn\text{-}ip}}$ of bounded-norm inner-product functionalities. Previous works on DDFE [29, 8] use the same functionality.

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

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

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

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

*All-or-Nothing Encapsulation (AoNE).* The notion of AoNE is a particular functionality of DDFE introduced by Chotard *et al.* [29]. In the transformation of [28, Section 5.2], AoNE appears under the name *Secret Sharing Layer (SSL)*. In [8], AoNE also serves as a building block for their FH-DDFE scheme and it is pointed out that function-hiding and standard security are the same for AoNE, for which there is no concept of keys.

**Definition 3 (All-or-Nothing Encapsulation).** *For $\lambda \in \mathbb{N}$, let $\mathsf{Tag}_\lambda = \mathsf{ID}_\lambda = \mathcal{R}_\lambda = \{0,1\}^{\mathrm{poly}(\lambda)}$, $\mathcal{K}_\lambda = \varnothing$, $\mathcal{M}_{\lambda,\mathrm{pub}} = 2^{\mathsf{ID}_\lambda} \times \mathsf{Tag}_\lambda$ and $\mathcal{M}_{\lambda,\mathrm{pri}} = \{0,1\}^L$ for a polynomial $L = L(\lambda) \colon \mathbb{N} \to \mathbb{N}$. The* all-or-nothing encapsulation *functionality $f^{\mathsf{aone}} = \{f^{\mathsf{aone}}_\lambda \colon \bigcup_{n\in\mathbb{N}} \mathsf{ID}^n_\lambda \times \bigcup_{n\in\mathbb{N}}(\mathsf{ID}_\lambda \times \mathcal{M}_\lambda)^n \to \mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ is defined via*

$$f^{\mathsf{dyn\text{-}ip}}_\lambda((i)_{i\in\mathcal{U}_K}, (i,m_i)_{i\in\mathcal{U}_M}) = \begin{cases} (x_i)_{i\in\mathcal{U}_M} & \textit{if condition } (*) \textit{ holds} \\ \perp & \textit{otherwise} \end{cases}$$

*for all $\lambda \in \mathbb{N}$, where condition $(*)$ holds if there exists $\mathsf{tag} \in \mathsf{Tag}_\lambda$ such that for each $i \in \mathcal{U}_M$, $m_i$ is of the form $(m_{i,\mathrm{pri}} := x_i, m_{i,\mathrm{pub}} := (\mathcal{U}_M, \mathsf{tag}))$.*

This means in particular that KeyGen is unnecessary and Dec works without taking secret keys as input because $\mathcal{U}_K$ can be the empty set. From [29], 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 independent of $|\mathcal{U}_M|$, but the security proof resorts to the ROM.

**Security.** We recall standard and function-hiding security for DDFE below.

**Definition 4** ($\boxed{\textbf{Function-Hiding}}$ **and** $\boxed{\textbf{Standard}}$ **Security).** *For a* ppt *adversary* $\mathcal{A}$ *against a DDFE scheme* $\mathcal{E}$ *for a functionality*

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

*we define the experiments* $\boxed{\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ *and* $\boxed{\mathbf{Exp}^{\mathsf{cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ *as shown in Figure 1. In the following all variables are indexed by* $\lambda$, *we omit the indices* $\lambda$ *for simplicity. The oracles* $\mathcal{O}\mathsf{HonestGen}$, $\mathcal{O}\mathsf{Enc}$, $\mathcal{O}\mathsf{KeyGen}$ *and* $\mathcal{O}\mathsf{Corrupt}$ *can be called in any order and any number of times.*[3] *We recall that for the queries to* $\mathcal{O}\mathsf{KeyGen}$ *and* $\mathcal{O}\mathsf{Enc}$, *namely* $\boxed{(i, k_i^{(0)}, k_i^{(1)})}$ $\boxed{(i, k_i)}$ *and* $(i, m_i^{(0)}, m_i^{(1)})$, *there are private parts* $k_{i,\text{pri}}^{\boxed{(b)}}, m_{i,\text{pri}}^{(b)}$ *and public parts* $k_{i,\text{pub}}^{\boxed{(b)}}, m_{i,\text{pub}}^{(b)}$ *in the keys as well as in the messages. We require* $m^{(0)}_{i,\text{pub}} = m^{(1)}_{i,\text{pub}} = m_{\text{pub}}$ $\boxed{\text{and } k^{(0)}_{i,\text{pub}} = k^{(1)}_{i,\text{pub}} = k_{\text{pub}}}$ *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* $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ $\boxed{\text{or there exists } (i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}}$ *such that* $i \in \mathcal{C}$ *and* $m_i^{(0)} \neq m_i^{(1)}$ $\boxed{\text{or } k_i^{(0)} \neq k_i^{(1)}}$ [4].
2. *For* $b \in \{0,1\}$, *there exist* $(i, k_i^{\boxed{(b)}})_{i\in\mathcal{U}_K} \in \bigcup_{n\in\mathbb{N}}(\mathsf{ID} \times \mathcal{K})^n$ *and* $(i, m_i^{(b)})_{i\in\mathcal{U}_M} \in \bigcup_{n\in\mathbb{N}}(\mathsf{ID} \times \mathcal{M})^n$ *such that*
   - $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ *and* $(i, \boxed{k_i^{(0)}, k_i^{(1)}}\ \boxed{k_i}) \in \mathcal{Q}_{\mathsf{KGen}}$ *for all* $i \in \mathcal{H}$,
   - $m_i^{(0)} = m_i^{(1)}$ $\boxed{\text{and } k_i^{(0)} = k_i^{(1)}}$ *for all* $i \in \mathcal{C}$, *and*
   - $f^{\mathsf{dyn}}((i, k_i^{\boxed{(0)}})_{i\in\mathcal{U}_K}, (i, m_i^{(0)})_{i\in\mathcal{U}_M}) \neq f^{\mathsf{dyn}}((i, k_i^{\boxed{(1)}})_{i\in\mathcal{U}_K}, (i, m_i^{(1)})_{i\in\mathcal{U}_M})$.

*Otherwise, we say that* $\mathcal{A}$ *is* admissible *w.r.t* $\mathcal{C}$, $\mathcal{Q}_{\mathsf{Enc}}$ *and* $\mathcal{Q}_{\mathsf{KGen}}$ *and write* $\mathsf{adm}(\mathcal{A}) = 1$. *We say that* $\mathcal{E}$ *is* $\boxed{\text{IND-secure}}$ *if for all* ppt *adversaries* $\mathcal{A}$,

$$\boxed{\mathbf{Adv}^{\mathsf{cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) := \left|\Pr\left[\mathbf{Exp}^{\mathsf{cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda) = 1\right] - \tfrac{1}{2}\right|} \ .$$

*Alternatively,* $\mathcal{E}$ *is said to be* $\boxed{\text{function-hiding}}$ *if for all* ppt *adversaries* $\mathcal{A}$,

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

*is negligible in* $\lambda$.

*Weaker Notions of Security.* One may define several 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 $\mathcal{C} = \{i\}$ to the oracle $\mathcal{O}\mathsf{Corrupt}$ must be submitted before $\mathsf{Initialize}$ 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 $(\mathsf{pk}_i, \mathsf{sk}_i)_{i\in\mathcal{C}}$ to $\mathcal{A}$. In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{stat\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

[3] W.l.o.g, we assume that each $i$ is queried at most once to $\mathcal{O}\mathsf{HonestGen}$ and $\mathcal{O}\mathsf{Corrupt}$, and that a query $\mathcal{O}\mathsf{Corrupt}(i)$ is always preceded by a query $\mathcal{O}\mathsf{HonestGen}(i)$.

[4] This condition was introduced in [27] then used in all other works on (D)MCFE [27, 45, 2, 1] and later on DDFE [29, 8]. A recent work [50] studies the relaxation that removes this condition for (D)MCFE, *i.e.* more attacks are considered admissible, and gives a provably secure DMCFE candidate computing inner products. We are not aware of any DDFE scheme in the literature which is proven secure under the stronger notion from [50].

```
$\underline{\mathsf{Initialize}(1^\lambda)}$:
    $b \xleftarrow{\$} \{0,1\}$
    $\mathcal{H}, \mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{KGen}} \leftarrow \varnothing$
    $\mathsf{pp} \leftarrow \mathsf{GSetup}(1^\lambda)$
    Return $\mathsf{pp}$
$\underline{\mathcal{O}\mathsf{HonestGen}(i)}$:
    $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{LSetup}(\mathsf{pp})$
    $\mathcal{H} \leftarrow \mathcal{H} \cup \{i\}$
    Return $\mathsf{pk}_i$
$\underline{\mathsf{Finalize}(b')}$:
    If $\mathsf{adm}(\mathcal{A}) = 1$, return $\beta \leftarrow (b' \overset{?}{=} b)$
    Else, return 0

$\underline{\mathcal{O}\mathsf{KeyGen}(i, \boxed{k_i^{(0)}, k_i^{(1)}}\ \boxed{k_i})}$:
    $\mathcal{Q}_{\mathsf{KGen}} \leftarrow \mathcal{Q}_{\mathsf{KGen}} \cup \{(i, \boxed{k_i^{(0)}, k_i^{(1)}}\ \boxed{k_i})\}$
    Return $\mathsf{dk}_i \leftarrow \mathsf{KeyGen}(\mathsf{sk}_i, k_i^{\boxed{(b)}})$
$\underline{\mathcal{O}\mathsf{Enc}(i, m_i^{(0)}, m_i^{(1)})}$:
    $\mathcal{Q}_{\mathsf{Enc}} \leftarrow \mathcal{Q}_{\mathsf{Enc}} \cup \{(i, m_i^{(0)}, m_i^{(1)})\}$
    Return $\mathsf{ct} \leftarrow \mathsf{Enc}(\mathsf{sk}_i, m_i^{(b)})$
$\underline{\mathcal{O}\mathsf{Corrupt}(i)}$:
    If $i \notin \mathcal{H}$: $(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathsf{LSetup}(\mathsf{pp})$
    $\mathcal{H} \leftarrow \mathcal{H} \setminus \{i\}$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{i\}$
    Return $\mathsf{sk}_i$
```

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

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 oracles $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{KeyGen}$ must be submitted before $\mathsf{Initialize}$ is called. In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{sel\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.
3. *Security against One Challenge:* $\mathbf{Exp}^{\mathsf{1chal\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ is as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$, except that the adversary must declare up front to $\mathsf{Initialize}$ additional public information for challenge messages and challenge keys $m_{\text{pub}}, k_{\text{pub}}$ so that:
   - if $(i, m_i^{(0)}, m_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ and $m_{i,\text{pub}}^{(0)} = m_{i,\text{pub}}^{(1)} \neq m_{\text{pub}}$, then $m_i^{(0)} = m_i^{(1)}$,
   - if $(i, k_i^{(0)}, k_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ and $k_{i,\text{pub}}^{(0)} = k_{i,\text{pub}}^{(1)} \neq k_{\text{pub}}$, then $k_i^{(0)} = k_i^{(1)}$.

   In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.
4. *Security against Complete Challenges:* Let $\mathsf{ID}$ be an identity space and $\mathsf{Tag}$ be a tag space. For a functionality $f^{\mathsf{dyn}}$ with $\mathcal{K}_{\lambda,\text{pub}} = \mathcal{M}_{\lambda,\text{pub}} = 2^{\mathsf{ID}_\lambda} \times \mathsf{Tag}_\lambda$, we consider a weakening of the security model with an additional constraint termed *complete challenges*. Specifically, the experiment $\mathbf{Exp}^{\mathsf{pos\text{-}fh}}_{\mathcal{E},f^{\mathsf{dyn}},\mathcal{A}}(1^\lambda)$ is the same as $\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},f^{\mathsf{dyn}},\mathcal{A}}(1^\lambda)$, except that we add the following condition *3* for $\mathsf{adm}(\mathcal{A}) = 0$:

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

   In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{pos\text{-}cpa}}_{\mathcal{E},f^{\mathsf{dyn}},\mathcal{A}}(1^\lambda)$.
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'. For $b \in \{0,1\}$, there exist $(i, k_i^{(b)})_{i \in \mathcal{U}_K} \in \bigcup_{n \in \mathbb{N}} (\mathsf{ID} \times \mathcal{K})^n$ as well as $(i, m_i^{(b)})_{i \in \mathcal{U}_M} \in \bigcup_{n \in \mathbb{N}} (\mathsf{ID} \times \mathcal{M})^n$ 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*
   - $f((i, k_i^{(0)})_{i \in \mathcal{U}_K}, (i, m_i^{(0)})_{i \in \mathcal{U}_M}) \neq f((i, k_i^{(0)})_{i \in \mathcal{U}_K}, (i, m_i^{(1)})_{i \in \mathcal{U}_M})$ OR
     $f((i, k_i^{(0)})_{i \in \mathcal{U}_K}, (i, m_i^{(1)})_{i \in \mathcal{U}_M}) \neq f((i, k_i^{(1)})_{i \in \mathcal{U}_K}, (i, m_i^{(1)})_{i \in \mathcal{U}_M})$ OR
     $f((i, k_i^{(0)})_{i \in \mathcal{U}_K}, (i, m_i^{(0)})_{i \in \mathcal{U}_M}) \neq f((i, k_i^{(1)})_{i \in \mathcal{U}_K}, (i, m_i^{(1)})_{i \in \mathcal{U}_M})$.

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

Lemma 5 uses a standard hybrid reduction to prove that in the weakly function-hiding setting (or for standard security), one-challenge security is equivalent to multi-challenge security. For completeness, the proof in the (more challenging) weakly function-hiding setting is given in Appendix A.6.

**Lemma 5.** *Let $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be a DDFE scheme for a functionality $f$. If $\mathcal{E}$ is* one-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*

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

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

The works of [47] and [4] present generic transformations that turn weakly function-hiding (multi-input) functional encryption schemes into full-fledged function-hiding schemes. A similar transformation, stated in Lemma 6, is also applicable in the case of IP-DDFE. The proof is given in Appendix A.7.

**Lemma 6.** *If there exists a weakly function-hiding DDFE scheme $\mathcal{E}$ for $f^{\mathsf{dyn\text{-}ip}}$, then there exists a (fully) function-hiding DDFE scheme $\mathcal{E}'$ for $f^{\mathsf{dyn\text{-}ip}}$. More precisely, for any* ppt *adversary $\mathcal{A}$, there exists a* ppt *algorithm $\mathcal{B}$ such that*

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

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

## 2.3 Decentralized Multi-Client Functional Encryption

The notion of *Decentralized Multi-Client Functional Encryption* (DMCFE) introduced in [27] can be identified as a special case of DDFE, where (1) the number of users is fixed in advanced by a (possibly interactive) global setup and there is no local setting up so that a new user can enter the system (2) the key of a user can be an *encryption key* to encrypt their private individual data (the user is a "client" in the terminology of [27]) or a *secret key* to generate a functional key component (the user is a "sender" in the terminology of [27]). Moreover, for efficiency, prior papers (such as [27, 28, 2, 1, 45, 29]) considered an additional *key combination* algorithm that, given $n$ functional key components $(\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i\in[n]}$ generated for the same tag $\mathsf{tag\text{-}f}$, outputs a succinct functional key $\mathsf{dk}_{\mathsf{tag\text{-}f}}$ which can be passed to decryption $\mathsf{Dec}(\mathsf{dk}_{\mathsf{tag\text{-}f}}, \mathbf{c})$. Without loss of generality, the DMCFE notion in this paper implicitly includes the key combination algorithm in the decryption algorithm and whenever we refer to other existing DMCFE schemes, they are syntactically understood as such. The formal definition of DMCFE that is used in this paper is given below.
Let $\{\mathsf{Tag}_\lambda\}_{\lambda\in\mathbb{N}}$, $\{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$, $\{\mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathsf{Param}_\lambda\}_{\lambda\in\mathbb{N}}$ be sequences of tag, domain, range and parameter spaces, respectively. We consider a function class $\mathcal{F} = \{\mathcal{F}_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$, where each $\mathcal{F}_{n,\lambda} = \{f_{n,\lambda,(y_1,\ldots,y_n)}\}_{(y_1,\ldots,y_n)}$ contains functions $f_{n,\lambda,(y_1,\ldots,y_n)} \colon \mathcal{D}^n_\lambda \to \mathcal{R}_\lambda$ described by their parameters $(y_1, \ldots, y_n) \in \mathsf{Param}^n_\lambda$.[5]

**Definition 7 (Decentralized Multi-Client Functional Encryption).** *A* 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 defined below:*

$\mathsf{Setup}(1^\lambda, 1^n)$: *This is a protocol between the senders that eventually generate their own secret keys $\mathsf{sk}_i$ and encryption keys $\mathsf{ek}_i$, as well as the public parameter $\mathsf{pp}$. We will assume that all the secret and encryption keys implicitly contain $\mathsf{pp}$.*

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

$\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i)$**:** *On input a secret key* $\mathsf{sk}_i$*, a tag* $\mathsf{tag\text{-}f} \in \mathsf{Tag}$*, and parameter* $y_i \in \mathsf{Param}_\lambda$*, this algorithm outputs a partial decryption key* $\mathsf{dk}_{\mathsf{tag\text{-}f},i}$*.*

$\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i)$**:** *On input an encryption key* $\mathsf{ek}_i$*, a tag* $\mathsf{tag}$ *and a message* $x_i \in \mathcal{D}_\lambda$*, 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 runs a key combination if necessary, then outputs an element* $d \in \mathcal{R}_\lambda$ *or a symbol* $\perp$*.*

The definitions of correctness and security (including all weaker notions) can be derived from that of DDFE (Definition 4). For completeness, we recall the explicit definitions for DMCFE in Appendix A.3. The function family $\mathcal{F}_n^{\mathsf{ip}}$ of bounded-norm inner-product functionalities with $n$ inputs is defined as follows.

**Definition 8 (Inner Product Functionality).** *For* $\lambda \in \mathbb{N}$*, let* $\mathcal{D}_\lambda = \mathsf{Param}_\lambda = [-B; B]^N$ *and* $\mathcal{R}_\lambda = \mathbb{Z}$*, where* $B = B(\lambda)$ *and* $N = N(\lambda)\colon \mathbb{N} \to \mathbb{N}$ *are polynomials. We define the* inner-product functionality $\mathcal{F}^{\mathsf{ip}} = \{\mathcal{F}^{\mathsf{ip}}_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$ *for* $\mathcal{F}^{\mathsf{ip}}_{n,\lambda} = \{f_{n,\lambda,(\mathbf{y}_1,\ldots,\mathbf{y}_n)} : \mathcal{D}_\lambda^n \to \mathcal{R}_\lambda\}_{(\mathbf{y}_1,\ldots,\mathbf{y}_n)\in\mathsf{Param}_\lambda^n}$ *as the family of functions* $f_{n,\lambda,(\mathbf{y}_1,\ldots,\mathbf{y}_n)}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i \rangle$*.*

## 3 Technical Overview

In this section, we give an overview about our generic conversion from DMCFE to DDFE as well as our FH-DMCFE scheme for inner products. The concrete construction of our FH-DMCFE heavily relies on the DPVS framework. We therefore divide this section in two parts. First, we discuss the main ideas behind our constructions on a high-level and compare them to existing schemes in the literature. Subsequently, we give a more technical overview about our FH-DMCFE scheme and its security proof in Section 3.1.

As a starting point, we attempt to follow the blueprint of Agrawal *et al.* [8]. They provide a construction of a FH-IP-DDFE scheme proven secure under *selective* key-generation and encryption queries as well as *static* corruption. Furthermore, their proof uses an additional constraint on the adversary termed *one key-label restriction.*[6] Our goal is to have a conversion that can be used independently of the function-hiding requirement, while achieving *adaptive* security for both encryption and key-generation queries under *static* corruption without relying on the one key-label restriction. The construction of [8] proceeds in two steps: Firstly, the authors build an FH-MCFE scheme, followed by a non-black-box transformation to DDFE. We recall their construction and explain our modifications that allow us to prove security in a stronger model. Furthermore, we identify specific structural properties that allow us to perform the second step in a black-box manner. By applying this generic conversion to our scheme built in step 1 as well as to other existing DMCFE schemes in the literature, we obtain several new DDFE constructions with previously unattained security guarantees.

**Step 1: Function-Hiding (D)MCFE.** We start by recalling that MCFE is a special case of DMCFE where a trusted authority is responsible for the generation of the functional decryption keys as well as the encryption keys $(\mathsf{ek}_i)_{i\in[n]}$ for the $n$ clients. The key held by the authority is called the master secret key msk.

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

*From Previous Works - The Function-Hiding* $\mathsf{MCFE}$ *of [8].* In their scheme, the encryption key $\mathsf{ek}_i$ of a client $i \in [n]$ consists of a master secret key $\mathsf{msk}_i$ of a (single-input) $\mathsf{FH\text{-}IPFE}$ scheme. The key-generating authority holds $\mathsf{msk} = (\mathsf{msk}_i)_{i\in[n]}$. Given a tuple $(i, \mathsf{tag}, \mathbf{x}_i)$, the encryption algorithm defines an extended vector of the form $\hat{\mathbf{x}}_i = (\mathbf{x}_i, \omega, \cdots)$, where $\omega = \mathsf{H}(\mathsf{tag})$ is a hash of the tag, and returns an encryption $\mathsf{ct}_i$ of $\hat{\mathbf{x}}_i$ under $\mathsf{msk}_i$. The dots $\cdots$ in the extended vector $\hat{\mathbf{x}}_i$ represent additional coordinates that are only used in the security proof and are 0 in the real scheme. Similarly, a functional decryption key $\mathsf{dk}_{\mathbf{y}}$ for a vector $\mathbf{y} = (\mathbf{y}_i)_{i\in[n]}$ is 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 $\mathsf{dk}_{\mathbf{y}} = (\mathsf{dk}_i)_{i\in[n]}$ where $\mathsf{dk}_i$ is a key for $\hat{\mathbf{y}}_i$ that is generated using $\mathsf{msk}_i$. Intuitively, decrypting $\mathsf{ct}_i$ with $\mathsf{dk}_i$ gives $\langle \mathbf{x}_i, \mathbf{y}_i\rangle + \omega s_i$. Since the value $s_i$ is secret, the term $\omega s_i$ serves as a mask that hides the partial inner product $\langle \mathbf{x}_i, \mathbf{y}_i\rangle$. On the other hand, if one has a ciphertext $\mathsf{ct}_i$ for each client $i \in [n]$ and all ciphertexts are encrypted w.r.t the same tag, then the sum of the partial decryptions gives $\sum_{i\in[n]}(\langle \mathbf{x}_i, \mathbf{y}_i\rangle + \omega s_i) = \sum_{i\in[n]}\langle \mathbf{x}_i, \mathbf{y}_i\rangle + \omega \cdot \sum_{i\in[n]} s_i = \sum_{i\in[n]}\langle \mathbf{x}_i, \mathbf{y}_i\rangle$, as $(s_i)_{i\in[n]}$ is a secret sharing of 0. The scheme is proven to be secure against selective adversaries that submit *all* oracle queries up front.

*Achieving Adaptive Security.* Before describing our changes from their scheme, it is instructive to analyze the difficulties that arise when attempting to build a simulator that is able to respond to encryption and key-generation queries adaptively. The admissibility for adversaries in the function-hiding security game for $\mathsf{MCFE}$ specifies *global* conditions. In the case of the inner-product functionality, they are 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, it is not excluded that *locally* at client $i$ we have $\langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)}\rangle \neq \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)}\rangle$. Therefore, during the security proof, switching from $(\mathbf{x}_i^{(0)})_i$ to $(\mathbf{x}_i^{(1)})_i$ will introduce a non-zero difference if the adversary tries to distinguish using the fact $\langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)}\rangle - \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)}\rangle \neq 0$. This non-zero difference caused by $\langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)}\rangle - \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)}\rangle$ must be compensated for the sake of indistinguishability between subsequent games in the security proof. For this reason, it happens that some queries $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ or $(\mathbf{y}_i^{(0)}, \mathbf{y}_i^{(1)})$ 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 keys or ciphertexts before they were even input to an oracle query.

In the $\mathsf{MCFE}$ of [8], this problem is solved by resorting to selective security. In this case, the simulator knows all inputs from the beginning and can use them whenever necessary. In contrast, we provide a concrete instantiation of the underlying single-input FE scheme for each $\mathsf{msk}_i$ based on DPVSes. If the simulator gets into a situation where it would have to use inputs $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(0)}, \mathbf{y}_i^{(1)})$ that have not yet been queried by the adversary, we make it guess them. Even though this guess degrades the probability of a successful efficient simulation by an exponential factor, it does not help the adversary because we design the games to have perfectly identical views, thanks to *information-theoretic* properties of the DPVS setting. In fact, this is the only point in our security proof where we crucially rely on DPVSes. Everywhere else, we could also employ a (black-box) single-input $\mathsf{FH\text{-}IPFE}$ scheme, as done in [8]. The technical details of our DPVS-based construction are quite involved and as mentioned above, we therefore describe them separately in Section 3.1.

*Security against Repeated Key Queries.* Even if the one key-label restriction (see footnote 6) is a constraint on the security model of the final $\mathsf{DDFE}$ scheme, the key for us to remove it is a modification in the underlying $\mathsf{MCFE}$ scheme. Therefore, we discuss this aspect at this point rather than below in Step 2 of the construction. Firstly, observe that both the syntax and the security definition of a function-hiding $\mathsf{DDFE}$ scheme (Definitions 1 and 4) are completely symmetric with respect to encryption and key generation. In particular, both ciphertexts and decryption keys are generated independently by each client, and an adversary is allowed to submit repeated queries (under the same tag) for both of them. On the other hand, there is an inherent asymmetry in the $\mathsf{MCFE}$ model. This consists in the fact that decryption keys are generated by a central authority rather than by the clients themselves, as is the case for ciphertexts. Moreover, the adversary can submit only *global* key queries on $(\mathbf{y}_i)_i$ containing for each sender $i$ a sub-vector $\mathbf{y}_i$. Intuitively, when viewing MCFE as a restricted particular case of DDFE (centralized setup and key generation with a fixed number of clients), these global queries $(\mathbf{y}_i)_i$ w.r.t the security model of MCFE correspond to a set of key queries w.r.t the security model of DDFE, namely exactly one query for each $\mathbf{y}_i$ under the same tag. Looking at the one key-label restriction, it can be noticed that this is exactly what is required therein. Phrased differently, the one key-label restriction imposed on the security model of DDFE by [8] seems to be exactly related to the structure of the MCFE model and the fact that [8] built their DDFE from MCFE. In order to lift this restriction, our approach to obtain a DDFE scheme whose security does not rely on this assumption is to *directly* build a scheme that is able to deal with repeated key queries. Then, using this newly built scheme, the DDFE scheme that is constructed below in Step 2 inherits the stronger security notion.

The natural generalization of MCFE that implements the concept of a *local* key query (*i.e.* a partial query $\mathbf{y}_i$ for only one sender $i$ at a time) is the DMCFE model. Therefore, our first part of the construction is not an MCFE but a DMCFE scheme which is secure against repeated key queries. As mentioned above, the desired security model is completely symmetric with respect to encryption and key generation. Therefore, it seems natural to aim for a symmetric construction, too. Inspecting the MCFE scheme of [8], it turns out that the asymmetry lies in the second component of the extended vectors $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$, *i.e.* the pair $(\omega, s_i)$ where $\omega$ is a hash of the ciphertext tag $\mathsf{tag}$ and $(s_i)_{i\in[n]}$ is a secret sharing of 0. Intuitively, the randomness of $s_i$ hides the partial inner product $\langle \mathbf{x}_i, \mathbf{y}_i\rangle$ even if the adversary submits encryption queries for several $\mathbf{x}_i$. By adding a flipped pair of tag and secret share to $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{y}}_i$, we can obtain the same properties for repeated key-generation queries. More precisely, if $\mu = \mathsf{H}(\mathsf{tag\text{-}f})$ denotes the hash of the function tag $\mathsf{tag\text{-}f}$ and $(t_i)_{i\in[n]}$ is another random secret sharing of 0, we set $\hat{\mathbf{x}}_i = (\mathbf{x}_i, \omega, t_i, \cdots)$ and $\hat{\mathbf{y}}_i = (\mathbf{y}_i, s_i, \mu, \cdots)$. The resulting scheme can be proven to be a FH-DMCFE whose security proof does not rely on the one key-label restriction anymore.

*Security against Incomplete Queries and Multiple Challenges.*

We recall from the above paragraph that our first step is constructing an FH-DMCFE scheme that is secure against repeated key queries. However, the security model of our initial construction suffers from two other restrictions that we call *one-challenge* security and the *complete-queries* constraint. (See conditions 3 and 4 in Definition 21 in the appendix for DMCFE, or conditions 3 and 4 in Definition 4 for the more general DDFE case). The complete-queries constraint is not particular to our construction and appears in many schemes in the literature, *e.g.* [8, 27, 1, 45]. On the other hand, the reason for the restriction to one-challenge security is due to technical subtleties in our security proof and does not occur in [8]. Intuitively, the above-mentioned information-theoretic arguments in the DPVS framework force us to consider queries one by one, whereas [8] can deal with all challenge queries at the same time in a uniform way. Fortunately, there exist conversions that remove both restrictions in a generic manner. For the sake of a more general result, we discuss them below in the DDFE setting (see paragraphs *Adaptive Security against Incomplete Queries.* and *Security against Multiple Challenge Queries*). Nevertheless, we emphasize that exactly the same transformations also work for (D)MCFE schemes and in the end our FH-DMCFE is secure against *multiple adaptive* challenge encryption queries and *multiple adaptive* challenge key-generation queries, where all queries are allowed to be incomplete and repetitive, under static corruption.

**Step 2: Transformation from (D)MCFE to DDFE.** The main difference between a DMCFE and a DDFE scheme for the inner product functionality is how the setup algorithm is executed. In the case of DMCFE, this is an interactive procedure with no easy way of adding new participants, while DDFE allows users to join the system at any time without interaction with the other users. Note that the dynamic set of users is also reflected in the DDFE version of the inner-product functionality which allows arbitrary support sets (Definition 2). This case is not considered in the DMCFE setting, as the set of clients is fixed up front (Definition 8). For the transformation from DMCFE to DDFE, it is therefore not enough to set up only one instance of the underlying DMCFE, but one needs to emulate an independent instance for each support, without interaction and for an exponential number of possible supports.

*From Previous Works - The Transformation of [8].* Let $\mathsf{ID}$ be some set of identities. The transformation of [8] uses two key ideas.
Firstly, recall that the encryption keys in their MCFE are independent master secret keys of a single-input function-hiding IPFE scheme. To generate them on demand and without interaction, each user $i$ in the DDFE is equipped with a key $K_i$ for a family of pseudorandom functions $\{F_K\}_K$. To obtain the encryption key w.r.t some support $\mathcal{U} \subseteq \mathsf{ID}$, user $i \in \mathcal{U}$ evaluates $F_{K_i}(\mathcal{U}) \to r_i$ and runs the setup algorithm of the single-input FE scheme with fixed random coins $r_i$. In this way it is guaranteed that user $i$ encrypts messages for the same support under the same key, but uses independent keys for different supports.
Secondly, the key-generation algorithm of the MCFE makes use of a random secret sharing $(s_i)_{i\in[n]}$ of 0, i.e. $\sum_{i\in[n]} s_i = 0$. For the transformation to DDFE, it arises the question of how such sharings can be generated in a decentralized and non-interactive manner. As a solution, the authors use a technique introduced in [29] under the name *decentralized sum* (DSum). This technique uses a clever interleaving of a non-interactive key exchange (NIKE) scheme and a PRF to generate a fresh sharing for each function.

*A Generic Conversion from DMCFE to DDFE.* We observe that both techniques mentioned in the previous paragraph can be applied in a much broader setting. Intuitively, we show that a DMCFE scheme can be lifted to DDFE whenever secret and encryption keys are of the form $\mathsf{sk}_i = (s_i, \widehat{\mathsf{sk}}_i)$ and $\mathsf{ek}_i = (s_i, \widehat{\mathsf{ek}}_i)$, where $(s_i)_i$ is a random secret sharing of 0 and $\widehat{\mathsf{sk}}_i$ and $\widehat{\mathsf{ek}}_i$ are generated independently of other users. As in [8], we use a DSum instance to compute a secret sharing $(s_i)_i$ and a PRF to generate the independent key components $\hat{\mathsf{sk}}_i$ and $\hat{\mathsf{ek}}_i$. In [29], it was shown that the DSum technique extends to any finite Abelian group $\mathbb{A}$. This allows us to apply the conversion to our FH-DMCFE scheme where the shares $s_i$ are vectors in $\mathbb{Z}_q^2$. (Recall that we use two scalar secret sharings, one in the keys and one in the ciphertexts). This gives us the first function-hiding IP-DDFE construction with adaptive security in the literature. Similarly, our conversion can be applied to the DMCFE scheme in [27] where shares $s_i$ are matrices in $\mathbb{Z}_q^{2\times 2}$. Furthermore, we observe that if the sum $s := \sum_i s_i$ is only needed at decryption time, then we are not limited to secret sharings of $s = 0_{\mathbb{A}}$, but $s$ can be chosen from a large variety of distributions. For instance, our technique extends to the case where $s$ is a sum of independent discrete Gaussian random variables. In this way, we are able to apply our conversion to the lattice-based DMCFE scheme of Libert *et al.* [45] which yields a IP-DDFE whose security is based solely on LWE in the standard model. Both properties were previously unattained. The details of our conversion are presented in Section 4, the instantiations are discussed in Sections B.3, B.4 and B.5.

*Security against Incomplete Queries.* To remove the complete-queries constraint (constraint 4 in Definition 4), previous works [29, 8] make use of a technique called *all-or-nothing encapsulation* (AoNE). Roughly, AoNE allows all parties of a 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 [29, 8], 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 $\mathsf{tag}$ (i.e. the global 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 global 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. Therefore, we think it is important to show that this AoNE layer indeed preserves adaptive security if the underlying scheme, which is only secure against complete queries, has this property. To our knowledge, the only known conversion that preserves adaptive security has been presented by Abdalla *et al.* [1]. The drawback of their construction is that global ciphertexts (*i.e.* one ciphertext for each client) grow quadratically in the number of clients. So, in conclusion, we show that the AoNE technique of [29] achieves the same security level as [1], while having global ciphertexts of only linear size if an appropriate instantiation for the AoNE scheme is used.

We present our result 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.2. Security is stated in Lemma 14. On an intuitive level, our simulator initially guesses whether or not the encryption queries for the challenge public input $k^*_{\text{pub}}$ (or $m^*_{\text{pub}}$) 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 argument 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. Afterwards, one can obtain security against multiple challenges by using the conversion described in the next paragraph.

We mention that a concurrent work by Shi and Vanjani [57] presents a similar conversion in the MCFE setting.

*Security against Multiple Challenge Queries.* It remains to discuss how a one-challenge DDFE scheme can be made resistant against multiple challenge queries. Firstly, observe that the equivalence of one-challenge and multi-challenge security in the standard setting (without function-hiding) is trivial. Indeed, the proof can be done by a sequence of hybrids over the different public inputs queried to the encryption oracle. This approach, however, 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 a mixed decryption. Therefore, we cannot change ciphertexts and keys one by one anymore. We solve this problem by first proving security in the weakly function-hiding setting (see Lemma 5). This model provides us exactly with the necessary guarantee for mixed decryptions, which allows a hybrid argument over public inputs to subsequently swap keys and ciphertexts. Afterwards, we apply another standard transformation that turns weakly function-hiding DDFE schemes for inner products into full-fledged function-hiding DDFE (see Lemma 6). Previous works [47, 4] presented that transformation for single-input and multi-input FE schemes.

## 3.1 Our Function-Hiding DMCFE for Inner Products

We give a simplified overview of our FH-DMCFE scheme in Section C.1 for the function class $\mathcal{F}_n^{\mathsf{ip}}$ defined in Definition 8. Specifically, we show *weakly function-hiding*, *one-challenge* security against *complete queries* under *static corruption*. As discussed earlier, the first three restrictions can be removed by applying several generic conversions. In this way, we obtain a FH-DMCFE for inner products whose only constraint on the security model is static corruption.

**Construction.** We work in the bilinear group setting $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q)$ assuming SXDH. Our construction relies on the notion of Dual Pairing Vector Spaces (DPVS). See Section 2.1 for basic definitions and Appendix A.2 for further details. During the setup, we sample two random secret sharings $(\tilde{s}_i)_{i\in[n]}, (\tilde{t}_i)_{i\in[n]}$ of 0 and an independent pair of dual bases $(\mathbf{B}_i = (\mathbf{b}_{i,j})_j, \mathbf{B}_i^* = (\mathbf{b}_{i,j}^*)_j)$ for each $i \in [n]$. Then we set $\mathsf{sk}_i = (\tilde{s}_i, \mathbf{B}_i^*)$ and $\mathsf{ek}_i = (\tilde{t}_i, \mathbf{B}_i)$. Each sender can use their secret key $\mathsf{sk}_i$ to independently generate a partial functional decryption key $\mathbf{d}_i$ for $\mathbf{y}_i \in \mathbb{Z}_q^N$, and each client can use their encryption key $\mathsf{ek}_i$ to independently compute a ciphertext $\mathbf{c}_i$ of their data $\mathbf{x}_i \in \mathbb{Z}_q^N$. We use two full-domain 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, where $\mathsf{Tag}$ denotes some set of tags. Given $\mathsf{tag}$ for encryption and $\mathsf{tag\text{-}f}$ for key generation, we denote $\omega \sim \mathsf{H}_1(\mathsf{tag})$ and $\mu \sim \mathsf{H}_2(\mathsf{tag\text{-}f})$ to indicate that $[\![\omega]\!]_1 = \mathsf{H}_1(\mathsf{tag})$ and $[\![\mu]\!]_2 = \mathsf{H}(\mathsf{tag\text{-}f})$. Recall from paragraph <u>*Security against Repeated Key Queries*</u> that we encrypt and generate keys for extended vectors of the form $\hat{\mathbf{x}}_i = (\mathbf{x}_i, \omega, t_i, \cdots)$ and $\hat{\mathbf{y}}_i = (\mathbf{y}_i, s_i, \mu, \cdots)$ where $(s_i)_i$ and $(t_i)_i$ are two secret sharings of 0. To generate $(s_i)_i$ in a decentralized manner, we use the secret sharing $(\tilde{s}_i)_i$ fixed during the (interactive) setup procedure and randomize it by setting $(s_i)_i := (\mu\tilde{s}_i)_i$. Under the DDH assumption in $\mathbb{G}_2$, such a multiple of $(\tilde{s}_i)_i$ cannot be distinguished from an independently randomly chosen secret sharing of 0. The same technique is applied for the decentralized generation of $(t_i)_i$, *i.e.* $(t_i)_i := (\mu\tilde{t}_i)_i$ for the secret sharing $(\tilde{t}_i)_i$ fixed during the setup. More specifically, the ciphertexts $\mathbf{c}_i$ and keys $\mathbf{d}_i$ in our FH-DMCFE have the following form:

$$\begin{array}{rc|c|c|c|c|cl}
\mathbf{c}_i = ( & \mathbf{x}_i & \omega \sim \mathsf{H}_1(\mathsf{tag}) & t_i = \tilde{t}_i\omega & 0 & \rho_i & \cdots & )_{\mathbf{B}_i} \\
\mathbf{d}_i = ( & \mathbf{y}_i & s_i = \tilde{s}_i\mu & \mu \sim \mathsf{H}_2(\mathsf{tag\text{-}f}) & \pi_i & 0 & \cdots & )_{\mathbf{B}_i^*}
\end{array}$$

where $\pi_i, \rho_i \xleftarrow{\$} \mathbb{Z}_q$ are random scalars, and the dots $\cdots$ represent additional coordinates that are only used in the security proof and are 0 in the real scheme.[7]

**Security.** We recall that the one-challenge restriction (constraint 3 in Definition 21) allows only one tag $\mathsf{tag}^*$ to the encryption oracle $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}^*, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ having $\mathbf{x}_i^{(0)} \neq \mathbf{x}_i^{(1)}$. Other tags $\mathsf{tag}_\ell \neq \mathsf{tag}^*$ and their corresponding inputs $(\mathbf{x}_{\ell,i}^{(0)}, \mathbf{x}_{\ell,i}^{(1)})$ to $\mathcal{O}\mathsf{Enc}$ are indexed by $\ell$ and it holds that $\mathbf{x}_{\ell,i}^{(0)} = \mathbf{x}_{\ell,i}^{(1)}$, so we can omit the superscript in this case. In the formal proof of Theorem 32, we add indices $j$ to denote repeated queries to the same client-tag pair. That is, the $j$-th query to $\mathcal{O}\mathsf{Enc}$ for client $i$ and tag $\mathsf{tag}^*$ (respectively $\mathsf{tag}_\ell$) is denoted by $(\mathbf{x}_i^{(0,j)}, \mathbf{x}_i^{(1,j)})$ (respectively $\mathbf{x}_{\ell,i}^{(j)}$). In the same manner, there exists only one $\mathsf{tag\text{-}f}^*$ queried to the key-generation oracle $\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}^*, \mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ having $\mathbf{y}^{(0)} \neq \mathbf{y}^{(1)}$, while for other $\mathsf{tag\text{-}f}_k \neq \mathsf{tag\text{-}f}^*$ it holds that $\mathbf{y}^{(0)} = \mathbf{y}^{(1)}$. We denote the $j$-th query to $\mathcal{O}\mathsf{DKeyGen}$ for client $i$ and tag $\mathsf{tag\text{-}f}^*$ (respectively $\mathsf{tag\text{-}f}_k$) by $(\mathbf{y}_i^{(0,j)}, \mathbf{y}_i^{(1,j)})$ (respectively $\mathbf{y}_{k,i}^{(j)}$). In this overview, for the ease of reading we omit the index $j$ when we do not differentiate the way

[7] Careful readers may have noticed that the encryption algorithm cannot compute $\omega \in \mathbb{Z}_q$, but only $\mathsf{H}_1(\mathsf{tag}) \to [\![\omega]\!]_1 \in \mathbb{G}_1$. Hence, to enable an efficient computation of $\mathbf{c}_i$, we actually include the scalar vectors $B_{i,N+1}, B_{i,N+2} \in \mathbb{Z}_q^N$ into $\mathsf{ek}_i$, as opposed to $\mathbf{b}_{i,N+1}, \mathbf{b}_{i,N+2} \in \mathbb{G}_1^N$. Similarly, to enable an efficient computation of $\mathbf{d}_i$, we include $B_{i,N+1}^*, B_{i,N+2}^* \in \mathbb{Z}_q^N$ instead of $\mathbf{b}_{i,N+1}^*, \mathbf{b}_{i,N+2}^* \in \mathbb{G}_2^N$ into $\mathsf{sk}_i$.

we handle repetitions for a specific $i$. To summarize, in the security game with the challenge bit $b \xleftarrow{\$} \{0,1\}$, the adversary obtains the following ciphertexts and keys:

$$\begin{array}{rl|c|c|c|c|cl}
\mathbf{c}_{\ell,i} = ( & \mathbf{x}_{\ell,i} & \omega_\ell \sim \mathsf{H}_1(\mathsf{tag}_\ell) & t_{\ell,i} = \tilde{t}_i\omega_\ell & 0 & \rho_{\ell,i} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{c}_i = ( & \mathbf{x}_i^{(b)} & \omega \sim \mathsf{H}_1(\mathsf{tag}^*) & t_i = \tilde{t}_i\omega & 0 & \rho_i & \cdots & )_{\mathbf{B}_i} \\
\mathbf{d}_i = ( & \mathbf{y}_i^{(b)} & s_i = \tilde{s}_i\mu & \mu \sim \mathsf{H}_2(\mathsf{tag}\text{-}\mathsf{f}^*) & \pi_i & 0 & \cdots & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = ( & \mathbf{y}_{k,i} & s_{k,i} = \tilde{s}_i\mu_k & \mu_k \sim \mathsf{H}_2(\mathsf{tag}\text{-}\mathsf{f}_k) & \pi_{k,i} & 0 & \cdots & )_{\mathbf{B}_i^*}
\end{array} \tag{1}$$

where $\pi_i, \pi_{k,i}, \rho_i, \rho_{\ell,i} \xleftarrow{\$} \mathbb{Z}_q$ are random scalars. Our goal is to switch $(\mathbf{x}_i^{(b)}, \mathbf{y}_i^{(b)})_i$ in $(\mathbf{c}_i, \mathbf{d}_i)_i$ to $(\mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)})_i$ so that the game does not depend on $b$ anymore, and the adversary's advantage becomes 0. We recall that variables with letters $\mathbf{c}$, $\mathbf{d}$, $\mathbf{x}$, $\mathbf{y}$, $\pi$ and $\rho$ implicitly carry an additional index $j$ for repetitions. On the other hand, 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 at setup time, hence they are the same across repetitions under the same tag. We denote $\mathcal{H} = [n] \setminus \mathcal{C}$ the set of honest clients. Note that the secret shares $s_i, t_i, s_{k,i}, t_{\ell,i}$ itself are hidden to the adversary for $i \in \mathcal{H}$. However, their sums

$$S \coloneqq \textstyle\sum_{i\in\mathcal{H}} s_i \qquad S_k \coloneqq \sum_{i\in\mathcal{H}} s_{k,i} \qquad T \coloneqq \sum_{i\in\mathcal{H}} t_i \qquad T_\ell \coloneqq \sum_{i\in\mathcal{H}} t_{\ell,i} \tag{2}$$

can be computed (*e.g.* via $S = -\sum_{i\in[n]\setminus\mathcal{H}} s_i$).

*Concrete Interpretation of the Admissibility.* The general conditions for an adversary $\mathcal{A}$ to be admissible (*i.e.* $\mathsf{adm}(\mathcal{A}) = 1$) are given in Definition 21. A step of vital importance in our proof is an interpretation of the admissibility for the concrete case of $\mathcal{F}_n^{\mathsf{ip}}$ which gives us:

- *Weakly Function-Hiding Security (Condition 2' in Definition 21).* In general, weakly function-hiding security requires that for all functions $(f^{(0)}, f^{(1)})$ and messages $(x_i^{(0)}, x_i^{(1)})_{i\in[n]}$ queried to $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Enc}$ under the same tag, it holds that $f^{(0)}(x_1^{(0)}, \ldots, x_n^{(0)}) = f^{(1)}(x_1^{(0)}, \ldots, x_n^{(0)}) = f^{(1)}(x_1^{(1)}, \ldots, x_n^{(1)})$. Translating this equation into our setting with one-challenge security and inner-product functionality, we obtain for all messages $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})_{i\in[n]}$ and function parameters $(\mathbf{y}_i^{(0)}, \mathbf{y}_i^{(1)})_{i\in[n]}$ queried w.r.t the challenge tags $\mathsf{tag}^*$ and $\mathsf{tag}\text{-}\mathsf{f}^*$ that

$$\textstyle\sum_{i\in[n]} \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)} \rangle = \sum_{i\in[n]} \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(1)} \rangle = \sum_{i\in[n]} \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)} \rangle \ . \tag{3}$$

  Note that for non-challenge tags, this equation holds trivially since left and right messages and function parameters must be the same.
- *Restricted Queries for Corrupted Clients (Condition 1 in Definition 21).* If a client $i \in [n]$ is corrupted by the adversary, then the challenge queries at $i$ must be on equal messages $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{(1)}$ and function parameters $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.

Thus, the honest clients $\mathcal{H} \subseteq [n]$ satisfy

$$\textstyle\sum_{i\in\mathcal{H}} \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)} \rangle \overset{(\bigstar)}{=} \sum_{i\in\mathcal{H}} \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(1)} \rangle \overset{(\blacktriangle)}{=} \sum_{i\in\mathcal{H}} \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)} \rangle \ . \tag{4}$$

*The Swapping Lemma.* The swapping lemma will be the centerpiece of our security proof. It features a number of interesting properties, which we believe make it of independent interest. Due to space limitations, we provide a separate overview of this lemma in Appendix C.4. The formal statement is presented in Lemma 34 and proven in Appendices C.5 and C.6. Here, we only state a simplified version.

**Lemma 9 (Swapping – Informal).** *Let $H, K, L, N, R, R_1, \ldots, R_K$ be public scalars and $(\mathbf{B}_i, \mathbf{B}_i^*)_{i\in[H]}$ be pairs of random dual bases that are kept secret. For each $i \in [H]$, consider public vectors $(\mathbf{u}_i, \mathbf{u}_{\ell,i}, \mathbf{u}'_{\ell,i}, \mathbf{v}_i^{(0)}, \mathbf{v}_i^{(1)}, \mathbf{v}_{k,i})_{k\in[K],\ell\in[L]} \in \mathbb{Z}_p^N$ so that $\sum_{i=1}^H \langle \mathbf{u}_i, \mathbf{v}_i^{(0)}\rangle = \sum_{i=1}^H \langle \mathbf{u}_i, \mathbf{v}_i^{(1)}\rangle$.*
*For secret $r, r_\ell, \rho_i, \rho_{\ell,i}, \pi_i, \pi_{k,i}, \sigma_i, \sigma_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ such that $\sum_{i=1}^H \sigma_i = R$ and $\sum_{i=1}^H \sigma_{k,i} = R_k$, for all $k \in [K]$, the following distributions are computationally indistinguishable under the* SXDH *assumption:*

$$\left\{\begin{array}{l}
\big((\mathbf{u}_{\ell,i}, \mathbf{u}'_{\ell,i}, r_\ell, 0, \rho_{\ell,i}, \cdots)_{\mathbf{B}_i}\big)_{i\in[H]}^{\ell\in[L]} \\
\big((\boxed{\mathbf{u}_i}, \boxed{\mathbf{0}}, r, 0, \rho_i, \cdots)_{\mathbf{B}_i}\big)_{i\in[H]} \\
\big((\mathbf{v}_i^{(0)}, \mathbf{v}_i^{(1)}, \sigma_i, \pi_i, 0, \cdots)_{\mathbf{B}_i^*}\big)_{i\in[H]} \\
\big((\mathbf{v}_{k,i}, \mathbf{v}_{k,i}, \sigma_{k,i}, \pi_{k,i}, 0, \cdots)_{\mathbf{B}_i^*}\big)_{i\in[H]}^{k\in[K]}
\end{array}\right\} \approx \left\{\begin{array}{l}
\big((\mathbf{u}_{\ell,i}, \mathbf{u}'_{\ell,i}, r_\ell, 0, \rho_{\ell,i}, \cdots)_{\mathbf{B}_i}\big)_{i\in[H]}^{\ell\in[L]} \\
\big((\boxed{\mathbf{0}}, \boxed{\mathbf{u}_i}, r, 0, \rho_i, \cdots)_{\mathbf{B}_i}\big)_{i\in[H]} \\
\big((\mathbf{v}_i^{(0)}, \mathbf{v}_i^{(1)}, \sigma_i, \pi_i^{(j)}, 0, \cdots)_{\mathbf{B}_i^*}\big)_{i\in[H]} \\
\big((\mathbf{v}_{k,i}, \mathbf{v}_{k,i}, \sigma_{k,i}, \pi_{k,i}, 0, \cdots)_{\mathbf{B}_i^*}\big)_{i\in[H]}^{k\in[K]}
\end{array}\right\}$$

Similar to above, all variables with letters $\mathbf{u}$, $\mathbf{v}$, $\pi$ and $\rho$ implicitly carry an additional index $j$ representing repetitions (*e.g.* we can have multiple $\mathbf{u}, \mathbf{u}'$ for a fixed $(\ell, i)$), but we omit it in this general overview for the sake of simplicity. We further note that this informal statement only reflects a "selective" variant of the real lemma. To prove adaptive security for the DMCFE scheme, we will use a stronger version, where all vectors with letters $\mathbf{u}$ and $\mathbf{v}$ can be chosen one by one.

We stress that this lemma is strictly more powerful than what can be done with black-box (single-input) FH-IPFE. If one replaces each pair $(\mathbf{B}_i, \mathbf{B}_i^*)$ with an independent FH-IPFE instance and encodes all vectors in $\mathbf{B}_i$ (resp. $\mathbf{B}_i^*$) in ciphertexts (resp. secret keys) of the $i$-th FH-IPFE instance, then one *cannot* conclude the above indistinguishability since the *local* inner products (at a single FH-IPFE instance) may be different. Indeed, the lemma does not require that $\langle \mathbf{u}_i, \mathbf{v}_i^{(0)}\rangle = \langle \mathbf{u}_i, \mathbf{v}_i^{(1)}\rangle$, so the admissibility is not satisfied and the FH-IPFE does not provide any security guarantees for this case.

*Finishing the Proof.* Being armed with Lemma 9, we are ready to tackle the security proof of our FH-DMCFE from Section C.1. First of all, since we consider only *static* corruptions, the simulator knows the corrupted clients from the very beginning and performs all changes only for honest $i \in \mathcal{H}$ whose $\mathsf{sk}_i = (\tilde{s}_i, \mathbf{B}_i^*)$ and $\mathsf{ek}_i = (\tilde{t}_i, \mathbf{B}_i)$ are never revealed to the adversary. We start by modifying the vectors as shown below (for details, see paragraph *Computational Basis Change* of Section A.2). Note that we use some of the auxiliary 0-coordinates represented by $\cdots$ in (1). For clarity, the changes are highlighted with $\boxed{\text{boxes}}$.

$$\begin{array}{rl|c|c|c|c|c|cl}
\mathbf{c}_{\ell,i} = ( & \mathbf{x}_{\ell,i} & \omega_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & \mathbf{0} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{c}_i = ( & \mathbf{x}_i^{(b)} & \omega & t_i & 0 & \rho_i & \mathbf{0} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{d}_i = ( & \mathbf{y}_i^{(b)} & s_i & \mu & \pi_i & 0 & \boxed{\mathbf{y}_i^{(1)}} & \cdots & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = ( & \mathbf{y}_{k,i} & s_{k,i} & \mu_k & \pi_{k,i} & 0 & \boxed{\mathbf{y}_{k,i}} & \cdots & )_{\mathbf{B}_i^*}
\end{array} \tag{5}$$

As a sanity check, observe that all vectors in $\mathbf{B}_i$ have only zeros in the coordinates corresponding to the newly introduced $\boxed{\mathbf{y}_i^{(1)}}$ and $\boxed{\mathbf{y}_{k,i}}$ in $\mathbf{B}_i^*$, so the inner products of vectors in $\mathbf{B}_i$ and $\mathbf{B}_i^*$ do not change. Now we are ready to use Lemma 9. We start with the challenge ciphertext $\mathbf{c}_i$. Our goal is to move the vector $\boxed{\mathbf{x}_i^{(b)}}$, which currently faces $\mathbf{y}_i^{(b)}$ in $\mathbf{d}_i$, to the position that faces $\mathbf{y}_i^{(1)}$ for all $i \in \mathcal{H}$. To this aim, we apply the swapping lemma by setting $\mathbf{u}_i := \mathbf{x}_i^{(b)}$, $\mathbf{u}_{\ell,i} := \mathbf{x}_{\ell,i}$, $\mathbf{u}'_{\ell,i} := \mathbf{0}$, $\mathbf{v}_i^{(0)} := \mathbf{y}_i^{(b)}$, $\mathbf{v}_i^{(1)} := \mathbf{y}_i^{(1)}$, $\mathbf{v}_{k,i} := \mathbf{y}_{k,i}$, $r := \omega$, $r_\ell := \omega_\ell$, $\sigma_i := s_i$, and $\sigma_{k,i} := s_{k,i}$. The constraints on $(\mathbf{u}_i, \mathbf{v}_i^{(b)})_i$ and $R, R_k$ are verified by equality ($\bigstar$) in (4) and by (2), respectively. Similarly, we perform a sequence of hybrids over distinct tags $\mathsf{tag}_\ell$ and apply the lemma to each $\boxed{\mathbf{x}_{\ell,i}}$, to finally arrive at:

$$\begin{array}{rl|c|c|c|c|c|cl}
\mathbf{c}_{\ell,i} = ( & \boxed{\mathbf{0}} & \omega_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & \boxed{\mathbf{x}_{\ell,i}} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{c}_{i} = ( & \boxed{\mathbf{0}} & \omega & t_{i} & 0 & \rho_{i} & \boxed{\mathbf{x}_{i}^{(b)}} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{d}_{i} = ( & \mathbf{y}_i^{(b)} & s_i & \mu & \pi_i & 0 & \mathbf{y}_i^{(1)} & \cdots & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = ( & \mathbf{y}_{k,i} & s_{k,i} & \mu_k & \pi_{k,i} & 0 & \mathbf{y}_{k,i} & \cdots & )_{\mathbf{B}_i^*}
\end{array} \quad (6)$$

Since the first $N$ coordinates of all vectors in $\mathbf{B}_i$ are equal to 0 now, we can also replace the corresponding positions in $\mathbf{d}_i$ and $\mathbf{d}_{k,i}$ with 0, relying again on computational basis changes:

$$\begin{array}{rl|c|c|c|c|c|cl}
\mathbf{d}_{i} = ( & \boxed{\mathbf{0}} & s_i & \mu & \pi_i & 0 & \mathbf{y}_i^{(1)} & \cdots & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = ( & \boxed{\mathbf{0}} & s_{k,i} & \mu_k & \pi_{k,i} & 0 & \mathbf{y}_{k,i} & \cdots & )_{\mathbf{B}_i^*}
\end{array} \quad (7)$$

Note that at this point all keys $\mathbf{d}_i$ and $\mathbf{d}_{k,i}$ are independent of the challenge bit $b$. Next we apply a sequence of basis changes symmetric to (5), (6) and (7), to achieve the same for the ciphertexts. We first use computational basis changes to introduce $\boxed{\mathbf{x}_i^{(1)}}$ and $\boxed{\mathbf{x}_{\ell,i}}$ in $\mathbf{c}_i$ and $\mathbf{c}_{\ell,i}$, using the fact that the first $N$ coordinates of all $\mathbf{d}$-vectors are 0. Then we apply Lemma 9 to swap $\boxed{\mathbf{y}_i^{(1)}}$ in the challenge key relying on the equalities (2) and ($\blacktriangle$) in (4). In the same vein, we go over all $\mathsf{tag}\text{-}\mathsf{f}_k$ to swap all $\boxed{\mathbf{y}_{k,i}}$. Eventually, we clean $\boxed{\boxed{\mathbf{x}_{\ell,i}}}$ and $\boxed{\boxed{\mathbf{x}_i^{(b)}}}$. At this point the game is independent of the challenge bit $b$, so the adversary's advantage is 0.

$$\begin{array}{rl|c|c|c|c|c|cl}
\mathbf{c}_{\ell,i} = ( & \boxed{\mathbf{x}_{\ell,i}} & \omega_\ell & t_{\ell,i} & 0 & \rho_{\ell,i} & \boxed{\boxed{\mathbf{0}}} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{c}_{i} = ( & \boxed{\mathbf{x}_{i}^{(1)}} & \omega & t_{i} & 0 & \rho_{i} & \boxed{\boxed{\mathbf{0}}} & \cdots & )_{\mathbf{B}_i} \\
\mathbf{d}_{i} = ( & \boxed{\mathbf{y}_i^{(1)}} & s_i & \mu & \pi_i & 0 & \mathbf{0} & \cdots & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{k,i} = ( & \boxed{\mathbf{y}_{k,i}} & s_{k,i} & \mu_k & \pi_{k,i} & 0 & \mathbf{0} & \cdots & )_{\mathbf{B}_i^*}
\end{array}$$

**Remark 10. (Usage of the ROM in FH-DMCFE/FH-DDFE)** In a concurrent work, Shi and Vanjani [57] 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 [20, 1, 58]. Using such a Cor-PRF, it is possible to generate a secret sharing of 0 in a decentralized way without interaction. In their scheme, each client embeds one share of such a secret sharing in their ciphertexts, similarly to our $(t_i)_i$. To tie the ciphertexts of the different clients together, their key components use common randomness in the coordinates facing $(t_i)_i$, similarly to our $\mu$. In this way, the $t_i$'s cancel out during decryption so as to enable correct decryption. In MCFE, the use of such common randomness in the keys does not pose a problem, as the functional key generation is centralized. Moving to the decentralized key generation of DMCFE, however, we do not know how to make key components agree on such a randomness without relying on ROs.

## 4 From DMCFE to DDFE

In this section, we generically build DDFE schemes from DMCFE, PRF and NIKE schemes. If the underlying DMCFE scheme satisfies the strong function-hiding security, so does the obtained DDFE. Furthermore, we present a black-box conversion that allows us to remove the incomplete-queries constraint in certain cases. The underlying DMCFE schemes are required to satisfy simple structural properties that are satisfied by our function-hiding IP-DMCFE in Section C.1 and various constructions in the literature [27, 45]. Even though all our instantiations are DMCFE schemes for the inner-product functionality, we believe it is interesting to note that our conversion does not require this. For instance, some works in the literature [5, 49] build functional encryption schemes for the inner-product functionality combined with fine-grained access control. Thanks to its general description, our conversion could also be applied to $\mathsf{DMCFE}$ schemes for this more complex functionality.

We start by formally defining the structural properties that a $\mathsf{DMCFE}$ scheme must meet to be compatible with our conversion to $\mathsf{DDFE}$. For a positive integer $n$ and a finite Abelian group $\mathbb{A}$, we define the set of sharings of $0_{\mathbb{A}}$ as $\mathcal{S}(n, \mathbb{A}) = \{(S_i)_{i\in[n]} \in \mathbb{A}^n : \sum_{i=1}^n S_i = 0_{\mathbb{A}}\}$. Overloading notation, we also denote by $\mathcal{S}(n, \mathbb{A}; r)$ a $\mathsf{ppt}$ algorithm that outputs a uniformly random element of $\mathcal{S}(n, \mathbb{A})$ using random coins $r$.

**Definition 11 (Dynamizability).** *Let $\mathbb{A}$ be a finite Abelian group. A $\mathsf{DMCFE}$ scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ is called* $\mathbb{A}$-dynamizable *if there exist $\mathsf{ppt}$ algorithms $\mathsf{SetupPP}$, $\mathsf{SetupUser}$ and functions $P_{\mathsf{sk}}, P_{\mathsf{ek}} \colon \{0,1\}^{\log|\mathbb{A}|} \to \{0,1\}^{\leq \log|\mathbb{A}|}$ so that for $(\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n; r)$, $(s_i)_{i\in[n]} \leftarrow \mathcal{S}(n, \mathbb{A}; r_{\mathcal{S}})$, $\widehat{\mathsf{pp}} \leftarrow \mathsf{SetupPP}(1^\lambda; r_0)$ and $\{(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i) \leftarrow \mathsf{SetupUser}(\widehat{\mathsf{pp}}; r_i)\}_{i\in[n]}$, the following distributions are equal:*

$$\left\{\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}\right\} = \left\{\widehat{\mathsf{pp}}, \left(P_{\mathsf{sk}}(s_i, \widehat{\mathsf{sk}}_i), P_{\mathsf{ek}}(s_i, \widehat{\mathsf{ek}}_i)\right)_{i\in[n]}\right\} \quad ,$$

*where the probability is taken over the random coins $r \overset{\$}{\leftarrow} \{0,1\}^{\mathrm{poly}(\lambda)}$ and $r_{\mathcal{S}}, r_0, \ldots, r_n \overset{\$}{\leftarrow} \{0,1\}^{\mathrm{poly}(\lambda)}$*

Next, we define the $\mathsf{DDFE}$ functionality $f^{\mathsf{dyn}}$ that we will obtain when plugging a $\mathsf{DMCFE}$ scheme for a function class $\mathcal{F}$ into our conversion.

**Definition 12 (Corresponding DDFE Functionality).** *Let $\{\mathcal{D}_\lambda\}_\lambda$, $\{\mathcal{R}_\lambda\}_\lambda$, $\{\mathsf{Param}_\lambda\}_\lambda$ be families of domain, output range and parameter spaces, respectively. Consider a $\mathsf{DMCFE}$ function classe $\mathcal{F} = \{\mathcal{F}_{n,\lambda}\}_{n,\lambda\in\mathbb{N}}$, where each $\mathcal{F}_{n,\lambda} = \{f_{n,\lambda,(y_1,\ldots,y_n)}\}_{(y_1,\ldots,y_n)}$ contains functions of the form $f_{n,\lambda,(y_1,\ldots,y_n)} \colon \mathcal{D}_\lambda^n \to \mathcal{R}_\lambda$. Furthermore, let $\{\mathsf{Tag}_\lambda\}_{\lambda\in\mathbb{N}}$, $\{\mathsf{ID}_\lambda\}_{\lambda\in\mathbb{N}}$, $\{\mathcal{K}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\{\mathcal{M}_\lambda\}_{\lambda\in\mathbb{N}}$ be families of tag, identity, key and message spaces, respectively, where $\mathsf{Tag}_\lambda = \mathsf{ID}_\lambda = \{0,1\}^*$, $\mathcal{K}_\lambda = \mathcal{K}_{\lambda,\mathrm{pri}} \times \mathcal{K}_{\lambda,\mathrm{pub}}$, $\mathcal{M}_\lambda = \mathcal{M}_{\lambda,\mathrm{pri}} \times \mathcal{M}_{\lambda,\mathrm{pub}}$, $\mathcal{K}_{\lambda,\mathrm{pri}} = \mathsf{Param}_\lambda$, $\mathcal{M}_{\lambda,\mathrm{pri}} = \mathcal{D}_\lambda$ and $\mathcal{K}_{\lambda,\mathrm{pub}} = \mathcal{M}_{\lambda,\mathrm{pub}} = 2^{\mathsf{ID}_\lambda} \times \mathsf{Tag}_\lambda$. The $\mathsf{DDFE}$ functionality $f^{\mathsf{dyn}} = \{f^{\mathsf{dyn}}_\lambda \colon \bigcup_{n\in\mathbb{N}}(\mathsf{ID}_\lambda \times \mathcal{K}_\lambda)^n \times \bigcup_{n\in\mathbb{N}}(\mathsf{ID}_\lambda \times \mathcal{M}_\lambda)^n \to \mathcal{R}_\lambda\}_{\lambda\in\mathbb{N}}$* corresponding to $\mathcal{F}$ *is defined via*

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

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

As an example, we observe that the $\mathsf{DDFE}$ inner-product functionality $f^{\mathsf{dyn\text{-}ip}}$ (Definition 2) corresponds to the $\mathsf{DMCFE}$ inner-product function class $\mathcal{F}^{\mathsf{ip}}$ (Definition 8). Our conversion employs an $\mathbb{A}$-dynamizable $\mathsf{DMCFE}$ scheme $\mathcal{E}' = (\mathcal{E}'.\mathsf{Setup}, \mathcal{E}'.\mathsf{DKeyGen}, \mathcal{E}'.\mathsf{Enc}, \mathcal{E}'.\mathsf{Dec})$, a NIKE $\mathcal{N} = (\mathcal{N}.\mathsf{Setup}, \mathcal{N}.\mathsf{SharedKey})$ and two families of pseudorandom functions $\{F_K\}_{K\in\mathcal{K}}$ and $\{F'_K\}_{K\in\mathcal{K}'}$, where the range of $\{F'_K\}_{K\in\mathcal{K}'}$ is a subset of $\mathbb{A}$. The details of our $\mathsf{DDFE}$ scheme $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ are given in Fig. 2. Decryption correctness is verified in Appendix B.1. Security is stated in the following theorem and also proven in Appendix B.1.

**Theorem 13.** *If $\mathcal{N}$ is an IND-secure NIKE scheme, $\{F_K\}_{K\in\mathcal{K}}$ and $\{F'_{K'}\}_{K'\in\mathcal{K}'}$ are families of pseudorandom functions and $\mathcal{E}'$ is a dynamizable (function-hiding) $\mathsf{DMCFE}$ scheme for a function*

$\mathsf{GSetup}(1^\lambda)$: On input the security parameter $1^\lambda$, run $\mathcal{E}'.\widehat{\mathsf{pp}} \leftarrow \mathcal{E}'.\mathsf{SetupPP}(1^\lambda)$ and $\mathcal{N}.\mathsf{pp} \leftarrow \mathcal{N}.\mathsf{Setup}(1^\lambda)$ and return $\mathsf{pp} := (\mathcal{E}'.\widehat{\mathsf{pp}}, \mathcal{N}.\mathsf{pp})$

$\mathsf{LSetup}(\mathsf{pp}, i)$: On input the public parameters $\mathsf{pp}$ 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 and return $\mathsf{dk}_i$ as follows:

$$\begin{aligned}
\forall j \in \mathcal{U}_K \setminus \{i\}: \quad & K'_{i,j} \leftarrow \mathcal{N}.\mathsf{SharedKey}(\mathcal{N}.\mathsf{sk}_i, \mathcal{N}.\mathsf{pk}_j) \\
s_i &= \sum_{j \in \mathcal{U}_K \setminus \{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_K) \\
(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i) &\leftarrow \mathcal{E}'.\mathsf{SetupUser}(\mathcal{E}'.\widehat{\mathsf{pp}}; F_{K_i}(\mathcal{U}_K)) \\
\mathsf{dk}_i &\leftarrow \mathcal{E}'.\mathsf{DKeyGen}(P_{\mathsf{sk}}(s_i, \widehat{\mathsf{sk}}_i), \mathsf{tag\text{-}f}, \mathbf{y}_i)
\end{aligned}$$

$\mathsf{Enc}(\mathsf{sk}_i, m)$: On input a secret key $\mathsf{sk}_i$ and $m = (\mathbf{x}_i, (\mathcal{U}_M, \mathsf{tag}))$ such that $i \in \mathcal{U}_M$, compute and return $\mathsf{ct}_i$ as follows:

$$\begin{aligned}
\forall j \in \mathcal{U}_M \setminus \{i\}: \quad & K'_{i,j} \leftarrow \mathcal{N}.\mathsf{SharedKey}(\mathcal{N}.\mathsf{sk}_i, \mathcal{N}.\mathsf{pk}_j) \\
s_i &= \sum_{j \in \mathcal{U}_M \setminus \{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_M) \\
(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i) &\leftarrow \mathcal{E}'.\mathsf{SetupUser}(\mathcal{E}'.\widehat{\mathsf{pp}}; F_{K_i}(\mathcal{U}_M)) \\
\mathsf{ct}_i &\leftarrow \mathcal{E}'.\mathsf{Enc}(P_{\mathsf{ek}}(s_i, \widehat{\mathsf{ek}}_i), \mathsf{tag}, \mathbf{x}_i)
\end{aligned}$$

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

Fig. 2: DDFE scheme $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ built from an $\mathbb{A}$-dynamizable DMCFE scheme $\mathcal{E}' = (\mathcal{E}'.\mathsf{Setup}, \mathcal{E}'.\mathsf{DKeyGen}, \mathcal{E}'.\mathsf{Enc}, \mathcal{E}'.\mathsf{Dec})$, a NIKE scheme $\mathcal{N} = (\mathcal{N}.\mathsf{Setup}, \mathcal{N}.\mathsf{SharedKey})$ and two PRFs $\{F_K\}_{K \in \mathcal{K}}$ and $\{F'_K\}_{K \in \mathcal{K}'}$

*class $\mathcal{F}$, then the DDFE scheme $\mathcal{E}$ in Fig. 2 for the functionality $f^{\mathsf{dyn}}$ corresponding to $\mathcal{F}$ is also (function-hiding) secure. 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*

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

*where* $\mathsf{xxx} \subseteq \{\mathsf{1chal}, \mathsf{pos}, \mathsf{sel}, \mathsf{wfh}, \mathsf{fh}\}$.

**Upgrading the Security Model.** By adding a layer of AoNE encryption on top of both ciphertexts and keys, we can make a DDFE scheme $\mathcal{E}$ secure against incomplete queries. While this technique is well known from [29, 8], we provide a new proof showing that the conversion preserves adaptive security if both $\mathcal{E}$ and the AoNE scheme are adaptively secure.

**Lemma 14.** *Assume there exist (1) a one-challenge (weakly function-hiding) DDFE scheme $\mathcal{E}^{\mathsf{pos}}$ for a functionality $f^{\mathsf{dyn}}$ 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 $f^{\mathsf{dyn}}$ that is even secure against incomplete queries. More precisely, for any* ppt *adversary $\mathcal{A}$, there exist* ppt *algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}yyy\text{-}cpa}}_{\mathcal{E}, f^{\mathsf{dyn}}, \mathcal{A}}(1^\lambda) \leq 6 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}xxx\text{-}yyy\text{-}cpa}}_{\mathcal{E}^{\mathsf{pos}}, f^{\mathsf{dyn}}, \mathcal{B}_1}(1^\lambda) + 6 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}cpa}}_{\mathcal{E}^{\mathsf{aone}}, f^{\mathsf{aone}}, \mathcal{B}_2}(1^\lambda) \ ,$$

*where* $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}\}$ *and* $\mathsf{yyy} \subseteq \{\mathsf{wfh}\}$.

Details are given in Appendix B.2. Subsequent to the conversion in Lemma 14, an application of Lemma 5 shows that the obtained construction is also secure against multiple challenges. However, we stress that the security proof of Lemma 14 crucially relies on the *one-challenge* setting. For instance, if $\mathcal{E}^{\mathsf{pos}}$ is $\{\mathsf{pos}, \mathsf{fh}\}$-secure, then our proof technique does not allow to remove the $\mathsf{pos}$ restriction directly. Instead, we first observe that $\{\mathsf{pos}, \mathsf{fh}\}$-security implies $\{\mathsf{1chal}, \mathsf{pos}, \mathsf{wfh}\}$-security. Then we can apply Lemma 14 to obtain $\{\mathsf{1chal}, \mathsf{wfh}\}$-security (which requires the $\mathsf{1chal}$ restriction), followed by an application of Lemma 5 to obtain $\{\mathsf{wfh}\}$-security (which requires the $\mathsf{wfh}$ restriction). Finally, if $\mathcal{E}^{\mathsf{pos}}$ is a DDFE scheme for the inner-product functionality $f^{\mathsf{dyn\text{-}ip}}$, then we can even upgrade $\mathsf{wfh}$ back to $\mathsf{fh}$ by using Lemma 6. We summarize our results in the following corollary.

**Corollary 15.** *Let $\varepsilon$ denote the empty string. Assume there exist (1) a dynamizable* xxx-yyy *DMCFE scheme $\mathcal{E}'$ for a function $f$, where* $\mathsf{xxx} \subseteq \{\mathsf{1chal}, \mathsf{pos}, \mathsf{stat}, \mathsf{sel}\}$ *and* $\mathsf{yyy} \in \{\varepsilon, \mathsf{wfh}, \mathsf{fh}\}$*, and (2) an AoNE scheme $\mathcal{E}^{\mathsf{aone}}$ whose message space contains the ciphertext space of $\mathcal{E}'$. Then there exists an* $\overline{\mathsf{xxx}}$-$\overline{\mathsf{yyy}}$ *DDFE scheme $\mathcal{E}$ for the same functionality $f$, where* $\overline{\mathsf{xxx}} = (\mathsf{xxx} \cap \{\mathsf{sel}\}) \cup \{\mathsf{stat}\}$*, and* $\overline{\mathsf{yyy}} = \varepsilon$ *if* $\mathsf{yyy} = \varepsilon$*,* $\overline{\mathsf{yyy}} = \mathsf{wfh}$ *if* $\mathsf{yyy} \in \{\mathsf{wfh}, \mathsf{fh}\}$ *and* $f \neq f^{\mathsf{dyn\text{-}ip}}$*, and* $\overline{\mathsf{yyy}} = \mathsf{fh}$ *if* $\mathsf{yyy} \in \{\mathsf{wfh}, \mathsf{fh}\}$ *and* $f = f^{\mathsf{dyn\text{-}ip}}$*,*

**Concrete Instantiations.** We can apply Corollary 15 to our FH-DMCFE presented below in Section C.1 as well as existing schemes in the literature [27, 45]. In this way, we obtain several IP-DDFE schemes for the functionality $f^{\mathsf{dyn\text{-}ip}}$ with previously unattained properties, which we highlight by a $\boxed{\text{frame}}$.

**Corollary 16.**
- Conversion of the FH-DMCFE of this work (Fig. 7). *There exists an FH-IP-DDFE scheme that is* $\boxed{\textit{adaptively secure}}$ *while allowing* $\boxed{\textit{repetitions for key-generation queries}}$*, against static corruption, under the* SXDH *assumption in the ROM. For details, see Section B.3.*
- Conversion of the DMCFE of [27]. *There exists a IP-DDFE scheme that is* $\boxed{\textit{adaptively secure}}$ *against static corruption under the* SXDH *assumption in the ROM. For details, see Section B.4.*
- Conversion of the DMCFE of [45]. *There exists a IP-DDFE scheme that is* $\boxed{\textit{adaptively secure}}$ *against static corruption under the* $\boxed{\text{LWE } \textit{assumption}}$ *in the* $\boxed{\textit{standard model}}$*. For details, see Section B.5.*

## Acknowledgments

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

## References

1. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Heidelberg (Dec 2019)
2. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Heidelberg (Apr 2019)
3. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (Mar / Apr 2015)
4. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Heidelberg (Aug 2018)
5. Abdalla, M., Catalano, D., Gay, R., Ursu, B.: Inner-product functional encryption with fine-grained access control. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 467–497. Springer, Heidelberg (Dec 2020)
6. Agrawal, S., Clear, M., Frieder, O., Garg, S., O'Neill, A., Thaler, J.: Ad hoc multi-input functional encryption. In: Vidick, T. (ed.) ITCS 2020. vol. 151, pp. 40:1–40:41. LIPIcs (Jan 2020)

7. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Heidelberg, Virtual Event (Aug 2021)
8. Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part II. LNCS, vol. 13043, pp. 224–255. Springer, Heidelberg (Nov 2021)
9. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption: Stronger security, broader functionality. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part I. LNCS, vol. 13747, pp. 711–740. Springer, Heidelberg (Nov 2022)
10. Agrawal, S., Libert, B., Stehlé, D.: Fully secure functional encryption for inner products, from standard assumptions. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 333–362. Springer, Heidelberg (Aug 2016)
11. Alwen, J., Krenn, S., Pietrzak, K., Wichs, D.: Learning with rounding, revisited - new reduction, properties and applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 57–74. Springer, Heidelberg (Aug 2013)
12. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part I. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (Aug 2015)
13. Ananth, P., Sahai, A.: Projective arithmetic functional encryption and indistinguishability obfuscation from degree-5 multilinear maps. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 152–181. Springer, Heidelberg (Apr / May 2017)
14. Attrapadung, N., Libert, B., de Panafieu, E.: Expressive key-policy attribute-based encryption with constant-size ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 90–108. Springer, Heidelberg (Mar 2011)
15. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 67–98. Springer, Heidelberg (Aug 2017)
16. Banerjee, A., Peikert, C., Rosen, A.: Pseudorandom functions and lattices. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 719–737. Springer, Heidelberg (Apr 2012)
17. Benhamouda, F., Bourse, F., Lipmaa, H.: CCA-secure inner-product functional encryption from projective hash functions. In: Fehr, S. (ed.) PKC 2017, Part II. LNCS, vol. 10175, pp. 36–66. Springer, Heidelberg (Mar 2017)
18. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015, Part I. LNCS, vol. 9452, pp. 470–491. Springer, Heidelberg (Nov / Dec 2015)
19. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th FOCS. pp. 171–190. IEEE Computer Society Press (Oct 2015)
20. Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H.B., Patel, S., Ramage, D., Segal, A., Seth, K.: Practical secure aggregation for privacy-preserving machine learning. In: Thuraisingham, B.M., Evans, D., Malkin, T., Xu, D. (eds.) ACM CCS 2017. pp. 1175–1191. ACM Press (Oct / Nov 2017)
21. Boneh, D., Franklin, M.K.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (Aug 2001)
22. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (Mar 2011)
23. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: Unbounded attributes and semi-adaptive security. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part III. LNCS, vol. 9816, pp. 363–384. Springer, Heidelberg (Aug 2016)
24. Castagnos, G., Laguillaumie, F., Tucker, I.: Practical fully secure unrestricted inner product functional encryption modulo p. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 733–764. Springer, Heidelberg (Dec 2018)
25. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (Feb 2007)
26. Chen, J., Lim, H.W., Ling, S., Wang, H., Wee, H.: Shorter IBE and signatures via asymmetric pairings. In: Abdalla, M., Lange, T. (eds.) PAIRING 2012. LNCS, vol. 7708, pp. 122–140. Springer, Heidelberg (May 2013)
27. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Heidelberg (Dec 2018)
28. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021 (2018), `https://eprint.iacr.org/2018/1021`
29. Chotard, J., Dufour-Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Dynamic decentralized functional encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 747–775. Springer, Heidelberg (Aug 2020)

30. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) 8th IMA International Conference on Cryptography and Coding. LNCS, vol. 2260, pp. 360–363. Springer, Heidelberg (Dec 2001)
31. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Cheng, C.M., Chung, K.M., Persiano, G., Yang, B.Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 164–195. Springer, Heidelberg (Mar 2016)
32. Datta, P., Dutta, R., Mukhopadhyay, S.: Strongly full-hiding inner product encryption. Theoretical Computer Science (2017). https://doi.org/https://doi.org/10.1016/j.tcs.2016.12.024, `https://www.sciencedirect.com/science/article/pii/S0304397516307526`
33. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the $k$-Linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 245–277. Springer, Heidelberg (Mar 2018)
34. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for Diffie-Hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (Aug 2013)
35. Freire, E.S.V., Hofheinz, D., Kiltz, E., Paterson, K.G.: Non-interactive key exchange. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 254–271. Springer, Heidelberg (Feb / Mar 2013)
36. Gay, R.: A new paradigm for public-key functional encryption for degree-2 polynomials. In: Kiayias, A., Kohlweiss, M., Wallden, P., Zikas, V. (eds.) PKC 2020, Part I. LNCS, vol. 12110, pp. 95–120. Springer, Heidelberg (May 2020)
37. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (May 2014)
38. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Predicate encryption for circuits from LWE. In: Gennaro, R., Robshaw, M.J.B. (eds.) CRYPTO 2015, Part II. LNCS, vol. 9216, pp. 503–523. Springer, Heidelberg (Aug 2015)
39. Gordon, S.D., Katz, J., Liu, F.H., Shi, E., Zhou, H.S.: Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774 (2013), `https://eprint.iacr.org/2013/774`
40. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 89–98. ACM Press (Oct / Nov 2006), available as Cryptology ePrint Archive Report 2006/309
41. Kim, S., Lewi, K., Mandal, A., Montgomery, H., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. In: Catalano, D., De Prisco, R. (eds.) SCN 18. LNCS, vol. 11035, pp. 544–562. Springer, Heidelberg (Sep 2018)
42. Kim, S., Kim, J., Seo, J.H.: A new approach for practical function-private inner product encryption. Cryptology ePrint Archive, Report 2017/004 (2017), `https://eprint.iacr.org/2017/004`
43. Langrehr, R.: On the multi-user security of lwe-based nike (2023), `https://eprint.iacr.org/2023/1401`
44. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (Feb 2010)
45. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Heidelberg (Dec 2019)
46. Lin, H.: Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017, Part I. LNCS, vol. 10401, pp. 599–629. Springer, Heidelberg (Aug 2017)
47. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In: Dinur, I. (ed.) 57th FOCS. pp. 11–20. IEEE Computer Society Press (Oct 2016)
48. Naor, M., Reingold, O.: Number-theoretic constructions of efficient pseudo-random functions. In: 38th FOCS. pp. 458–467. IEEE Computer Society Press (Oct 1997)
49. Nguyen, K., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with fine-grained access-control. In: Asiacrypt '22. Springer-Verlag (2022), `https://ia.cr/2022/215`
50. Nguyen, K., Phan, D.H., Pointcheval, D.: Optimal security notion for decentralized multi-client functional encryption. In: 21st International Conference on Applied Cryptography and Network Security. Springer-Verlag (2023), `https://ia.cr/2023/435`
51. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (Aug 2010)
52. Okamoto, T., Takashima, K.: Adaptively attribute-hiding (hierarchical) inner product encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (Apr 2012)
53. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (Dec 2012)
54. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., De Capitani di Vimercati, S., Syverson, P.F. (eds.) ACM CCS 2007. pp. 195–203. ACM Press (Oct 2007)

55. Sahai, A., Waters, B.R.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (May 2005)
56. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO'84. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (Aug 1984)
57. Shi, E., Vanjani, N.: Multi-Client Inner Product Encryption: Function-Hiding Instantiations Without Random Oracles. In: International Conference on Practice and Theory of Public-Key Cryptography (PKC) (2023), `https://eprint.iacr.org/2023/615`
58. Shi, E., Wu, K.: Non-interactive anonymous router. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part III. LNCS, vol. 12698, pp. 489–520. Springer, Heidelberg (Oct 2021)
59. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Bishop, M., Nascimento, A.C.A. (eds.) ISC 2016. LNCS, vol. 9866, pp. 408–425. Springer, Heidelberg (Sep 2016)
60. Ünal, A.: Impossibility results for lattice-based functional encryption schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part I. LNCS, vol. 12105, pp. 169–199. Springer, Heidelberg (May 2020)
61. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (Aug 2009)

## A Supporting Materials – Section 2

### A.1 Hardness Assumptions

We state the assumptions needed for our constructions.

**Definition 17 (Decisional Diffie-Hellman).** *In a cyclic group $\mathbb{G}$ of prime order $q$, the* Decisional Diffie-Hellman (DDH) *problem is to distinguish the distributions*

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

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

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

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

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

**Definition 19 (Symmetric External Diffie-Hellman).** *In the bilinear setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{t}}, g_1, g_2, g_{\mathrm{t}}, \mathbf{e}, q)$, the* Symmetric eXternal Diffie-Hellman (SXDH) *assumption makes the* DDH *assumption in both $\mathbb{G}_1$ and $\mathbb{G}_2$.*

We denote by $\mathsf{GGen}(1^\lambda)$ the algorithm that on input the security parameter outputs a description of a bilinear group $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_{\mathrm{t}}, g_1, g_2, g_{\mathrm{t}}, \mathbf{e}, q)$ that satisfies the SXDH assumption.

**Definition 20 (Learning with Errors).** *Let $\alpha : \mathbb{N} \to (0, 1)$ and $m \geq n \geq 1$, $q \geq 2$ be functions of a security parameter $\lambda \in \mathbb{N}$. We write vectors as* column *vectors. The* Learning with Errors (LWE) *problem consists in distinguishing between the distributions $(\mathbf{A}, \mathbf{s}^\top \mathbf{A} + \mathbf{e}^\top)$ and $U(\mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m)$, where $\mathbf{A} \sim U(\mathbb{Z}_q^{n \times m})$, $\mathbf{s} \sim U(\mathbb{Z}_q^n)$ and $\mathbf{e} \sim D_{\mathbb{Z}^m, \alpha q}$. For a* ppt *algorithm $\mathcal{A}\colon \mathbb{Z}_q^{n \times m} \times \mathbb{Z}_q^m \to \{0, 1\}$, we define*

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

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

We require that $\alpha \geq 2\sqrt{n}/q$ for the reduction from worst-case lattice problems and refer the readers to, *e.g.*, [23] for more details.

### A.2 Dual Pairing Vector Spaces

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

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

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

$$(x_1,\dots,x_N)_{\mathbf{B}} = ((x_1,\dots,x_N)\cdot B)_{\mathbf{U}}\ ;\ (y_1,\dots,y_N)_{\mathbf{U}} = \left((y_1,\dots,y_N)\cdot B^{-1}\right)_{\mathbf{B}} \ . \tag{8}$$

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

$$(x_1,\dots,x_N)_{\mathbf{B}^*} = \left((x_1,\dots,x_N)\cdot B'\right)_{\mathbf{U}^*}\ ;\ (y_1,\dots,y_N)_{\mathbf{U}^*} = \left((y_1,\dots,y_N)\cdot B^\top\right)_{\mathbf{B}^*} \ . \tag{9}$$

It can be checked that $(\mathbf{B},\mathbf{B}^*)$ remains a pair of dual orthogonal bases. When we consider a basis change $\mathbf{B} = B\cdot\mathbf{U}$, if $B = (b_{i,j})_{i,j}$ affects only a subset $J \subseteq [N]$ of indices in the representation w.r.t basis $\mathbf{U}$, we will write $B$ as the square block containing $(b_{i,j})_{i,j}$ for $i,j \in J$ and implicitly the entries of $B$ outside this block are taken from the identity matrix $I_N$.
The basis changes are particularly useful in our security proofs. Intuitively these changes constitute a transition from a hybrid $\mathsf{G}$ having vectors expressed in $(\mathbf{U},\mathbf{U}^*)$ to the next hybrid $\mathsf{G}_{\mathsf{next}}$ having vectors expressed in $(\mathbf{B},\mathbf{B}^*)$. We focus on two types of basis changes, which are elaborated below. For simplicity, we consider dimension $N = 2$:

*Formal Basis Change:* We change $(\mathbf{U},\mathbf{U}^*)$ into $(\mathbf{B},\mathbf{B}^*)$ using

$$B := \begin{bmatrix}1 & 0\\ 1 & 1\end{bmatrix}_{1,2} \qquad B' := \left(B^{-1}\right)^\top = \begin{bmatrix}1 & -1\\ 0 & 1\end{bmatrix}_{1,2}$$
$$\mathbf{B} = B\cdot\mathbf{U} \qquad \mathbf{B}^* = B'\cdot\mathbf{U}^* \ .$$

We use this type in situations such as: in $\mathsf{G}$ we have vectors *all* of the form $(x_1,0)_{\mathbf{U}}, (y_1,0)_{\mathbf{U}^*}$ and we want to go to $\mathsf{G}_{\mathsf{next}}$ having *all* of the form $(x_1,0)_{\mathbf{B}}, (y_1,\boxed{y_1})_{\mathbf{B}^*}$. The simulator writes *all* vectors $(x_1,0)_{\mathbf{U}}, (y_1,0)_{\mathbf{U}^*}$ in $(\mathbf{U},\mathbf{U}^*)$ and under this basis change they are written into

$$(x_1,\ 0)_{\mathbf{U}} = (x_1-0,\ 0)_{\mathbf{B}} = (x_1,\ 0)_{\mathbf{B}};\quad (y_1,\ 0)_{\mathbf{U}^*} = (y_1,\ 0+y_1)_{\mathbf{B}^*} = (y_1,\ y_1)_{\mathbf{B}^*}$$

following the calculations in (8) and (9). The products between two dual vectors are invariant, *all* vectors are formally written from $(\mathbf{U},\mathbf{U}^*)$ (corresponding to $\mathsf{G}$) to $(\mathbf{B},\mathbf{B}^*)$ (corresponding to $\mathsf{G}_{\mathsf{next}}$), the adversary's view over the vectors is thus identical from $\mathsf{G}$ to $\mathsf{G}_{\mathsf{next}}$. In particular, this is a kind of *information-theoretic property* of DPVS by basis changing that we exploit to have identical hybrids' hop in the security proof.

*Computational Basis Change:* Given an instance of a computational problem, *e.g.* $[\![(a,b,c)]\!]_1$ of $\mathsf{DDH}$ in $\mathbb{G}_1$ where $c - ab = 0$ or $\delta \xleftarrow{\$} \mathbb{Z}_q$, we change $(\mathbf{U},\mathbf{U}^*)$ into $(\mathbf{B},\mathbf{B}^*)$ using

$$B := \begin{bmatrix}1 & 0\\ a & 1\end{bmatrix}_{1,2} \qquad B' := \left(B^{-1}\right)^\top = \begin{bmatrix}1 & -a\\ 0 & 1\end{bmatrix}_{1,2}$$
$$\mathbf{B} = B\cdot\mathbf{U} \qquad \mathbf{B}^* = B'\cdot\mathbf{U}^* \ .$$

One situation where this type of basis change can be useful is: in $\mathsf{G}$ we have *some* target vectors of the form $(0, \mathsf{rnd})_{\mathbf{U}}$, where $\mathsf{rnd} \xleftarrow{\$} \mathbb{Z}_q$ is a random scalar, together with other $(z_1, z_2)_{\mathbf{U}}$, and *all* the dual is of the form $(0,\ y_2)_{\mathbf{U}^*}$. We want to go to $\mathsf{G}_{\mathsf{next}}$ having $(\boxed{\widetilde{\mathsf{rnd}}},\ \mathsf{rnd})_{\mathbf{B}}$ masked by some randomness $\widetilde{\mathsf{rnd}} \xleftarrow{\$} \mathbb{Z}_q$, while keeping $(0,\ y_2)_{\mathbf{B}^*}$. Because $[\![a]\!]_1$ is given, the simulator can simulate vectors $(z_1, z_2)_{\mathbf{U}}$ directly in $\mathbf{B}$ using $[\![a]\!]_1$ as well as the known coordinates $z_1, z_2$. The basis change will be employed for the simulation of target vectors:

$$(c,\ b)_{\mathbf{U}} + (0, \mathsf{rnd})_{\mathbf{B}} = (c - a \cdot b,\ \mathsf{rnd} + b)_{\mathbf{B}};$$
$$(0,\ y_2)_{\mathbf{U}^*} = (0,\ y_2 + a \cdot 0)_{\mathbf{B}^*} = (0,\ y_2)_{\mathbf{B}^*}$$

where *all* vectors in $\mathbf{B}^*$ must be written first in $\mathbf{U}^*$, since we do not have $[\![a]\!]_2$, to see how the basis change affects them. Using the basis change we simulate those target vectors by $(c - a \cdot b,\ \mathsf{rnd} + b)_{\mathbf{B}}$ with $\mathsf{rnd}$ implicitly being updated to $\mathsf{rnd} + b$, the uninterested $(z_1, z_2)_{\mathbf{B}}$ are simulated correctly in $\mathbf{B}$, meanwhile the dual vectors $(0,\ y_2)_{\mathbf{B}^*}$ stays the same. Depending on the DDH instance, if $c - ab = 0$ the target vectors are in fact $(0,\ \mathsf{rnd})_{\mathbf{B}}$ and we are simulating $\mathsf{G}$, else $c - ab = \delta \xleftarrow{\$} \mathbb{Z}_q$ the target vectors are simulated for $\mathsf{G}_{\mathsf{next}}$ and $\widetilde{\mathsf{rnd}} := \delta$. Hence, under the hardness of DDH in $\mathbb{G}_1$, a computationally bounded adversary cannot distinguish its views in the hybrids' hop from $\mathsf{G}$ to $\mathsf{G}_{\mathsf{next}}$.

We remark that the basis changes will modify basis vectors and for the indistinguishability to hold, perfectly in *formal* change and computationally in *computational* changes, all impacted basis vectors must not be revealed to the adversary.

**Additional Notations.** Any $\mathbf{x} = [\![(m_1, \dots, m_N)]\!]_1 \in \mathbb{G}_1^N$ is identified as the vector $(m_1, \dots, m_N) \in \mathbb{Z}_q^N$. There is no ambiguity because $\mathbb{G}_1$ is a cyclic group of order $q$ prime. The $\mathbf{0}$-vector is $\mathbf{0} = [\![(0, \dots, 0)]\!]_1$. 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 $t \cdot \mathbf{x} := [\![t \cdot (m_1, \dots, m_N)]\!]_1$, where $t \in \mathbb{Z}_q$ and $\mathbf{x} = [\![(m_1, \dots, m_N)]\!]_1$. The additive inverse of $\mathbf{x} \in \mathbb{G}_1^N$ is defined to be $-\mathbf{x} := [\![(-m_1, \dots, -m_N)]\!]_1$. The canonical basis $\mathbf{A}$ of $\mathbb{G}_1^N$ consists of $\mathbf{a}_1 := [\![(1, 0 \dots, 0)]\!]_1, \mathbf{a}_2 := [\![(0, 1, 0 \dots, 0)]\!]_1, \dots, \mathbf{a}_N := [\![(0, \dots, 0, 1)]\!]_1$. By convention the writing $\mathbf{x} = (m_1, \dots, m_N)$ concerns the canonical basis $\mathbf{A}$.

## A.3 Function-Hiding Decentralized Multi-Client FE

This section complements Section 2.3 with additional definitions.

**Correctness.** $\mathcal{E}$ is *correct* if for all $\lambda, n \in \mathbb{N}$, $(x_1, \dots, x_n) \in \mathcal{D}_\lambda^n$, $f_{n,\lambda,(y_1,\dots,y_n)} \in \mathcal{F}_{n,\lambda}$ having parameters $(y_1, \dots, y_n) \in \mathsf{Param}_\lambda^n$, and for any $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}_\lambda$, we have

$$\Pr\left[ d = f_{n,\lambda,(y_1,\dots,y_n)}(x_1, \dots, x_n) \;\middle|\; \begin{array}{l} (\mathsf{pp}, (\mathsf{sk}_i)_{i \in [n]}, (\mathsf{ek}_i)_{i \in [n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \forall i \in [n] \colon \mathsf{dk}_{\mathsf{tag\text{-}f}, i} \leftarrow \mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i) \\ \forall i \in [n] \colon \mathsf{ct}_{\mathsf{tag}, i} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i) \\ d := \mathsf{Dec}((\mathsf{dk}_{\mathsf{tag\text{-}f}, i})_{i \in [n]}, (\mathsf{ct}_{\mathsf{tag}, i})_{i \in [n]}) \end{array} \right] = 1$$

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

**Security.** We define function-hiding and standard security for DMCFE. In the seminal work by Chotard *et al.* [27] and its follow-up study [29], the security notion does not cover the function-hiding requirement for DMCFE or its more general sibling DDFE. Until recently, the work by Agrawal *et al.* [8] abstracted out DMCFE into the notion of *Multi-Party Functional Encryption* (MPFE). The authors of [8] 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 for convenience of the reader, we introduce the detailed function-hiding security for DMCFE, without going through all the abstraction of MPFE nor of DDFE.

**Definition 21** ($\boxed{\textbf{Function-Hiding}}$ **and** $\dbox{\textbf{Standard}}$ **Security).** *For a DMCFE scheme $\mathcal{E}$, a function class $\mathcal{F} = \{\mathcal{F}_{n,\lambda}\}_{n,\lambda}$ and a* ppt *adversary $\mathcal{A}$ we define the experiments* $\boxed{\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ $\dbox{\mathbf{Exp}^{\mathsf{cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ *as shown in Figure 3 and set $\mathcal{H} := [n] \setminus \mathcal{C}$. 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 $\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{KGen}}$, denoted by $\mathsf{adm}(\mathcal{A}) = 0$, if either one of the following holds:*

1. *There exists a tuple $(i, \mathsf{tag}, x_i^{(0)}, x_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ $\boxed{\text{or } (i, \mathsf{tag\text{-}f}, y_i^{(0)}, y_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}}$ such that $i \in \mathcal{C}$ and $x_i^{(0)} \neq x_i^{(1)}$*[8]*,* $\boxed{\text{or } y_i^{(0)} \neq y_i^{(1)}}$.
2. *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 functions* $\boxed{f^{(0)}_{n,\lambda,(y_1^{(0)},\ldots,y_n^{(0)})}, f^{(1)}_{n,\lambda,(y_1^{(1)},\ldots,y_n^{(1)})} \in \mathcal{F}}$ $\dbox{f_{n,\lambda,y_1,\ldots,y_n} \in \mathcal{F}}$ *having parameters* $\boxed{(y_i^{(0)}, y_i^{(1)})_{i\in[n]}}$ $\dbox{(y_i)_{i\in[n]}}$ *such that*
   - $(i, \mathsf{tag}, x_i^{(0)}, x_i^{(1)}) \in \mathcal{Q}_{\mathsf{Enc}}$ *and* $(i, \mathsf{tag\text{-}f}, \boxed{y_i^{(0)}, y_i^{(1)}} \dbox{y_i}) \in \mathcal{Q}_{\mathsf{KGen}}$ *for all $i \in \mathcal{H}$,*
   - $x_i^{(0)} = x_i^{(1)}$ $\boxed{\text{and } y_i^{(0)} = y_i^{(1)}}$ *for all $i \in \mathcal{C}$, and*
   - $f^{\boxed{(0)}}_{n,\lambda,y_1^{(0)},\ldots,y_n^{(0)}}(x_1^{(0)}, \ldots, x_n^{(0)}) \neq f^{\boxed{(1)}}_{n,\lambda,y_1^{(1)},\ldots,y_n^{(1)}}(x_1^{(1)}, \ldots, x_n^{(1)})$.

*Otherwise, we say that $\mathcal{A}$ is* admissible *w.r.t $\mathcal{C}$, $\mathcal{Q}_{\mathsf{Enc}}$ and $\mathcal{Q}_{\mathsf{KGen}}$ and write $\mathsf{adm}(\mathcal{A}) = 1$. We call $\mathcal{E}$* $\boxed{\text{function-hiding } \dbox{\text{IND-secure}}}$ *if for all* ppt *adversaries $\mathcal{A}$,*

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

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

*is negligible in $\lambda$.*

**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. In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{stat\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.
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 oracles $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Enc}$ must be submitted before $\mathsf{Initialize}$ is called. In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{sel\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.
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 $(i, \mathsf{tag\text{-}f}, y_i^{(0)}, y_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ and $\mathsf{tag\text{-}f} \neq \mathsf{tag\text{-}f}^*$, then $y_i^{(0)} = y_i^{(1)}$.

[8] We refer to Footnote 4 for a discussion on this condition.

```
$\underline{\mathsf{Initialize}(1^\lambda)\text{:}}$
$\mathcal{C}, \mathcal{Q}_{\mathsf{Enc}}, \mathcal{Q}_{\mathsf{KGen}} \leftarrow \varnothing;\ b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pp}, (\mathsf{sk}_i)_{i\in[n]}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n)$
Return $\mathsf{pp}$

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

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

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

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

Fig. 3: Security games $\boxed{\mathbf{Exp}^{\mathsf{fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ and $\dbox{\mathbf{Exp}^{\mathsf{cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)}$ for Definition 21

In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

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 *3* 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 $i \in \mathcal{H}$. In the same vein, we can define experiment $\mathbf{Exp}^{\mathsf{pos\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$.

5. *Weak* $\boxed{\textit{Function-Hiding}}$*:* 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'. 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* $\boxed{f^{(0)}_{n,\lambda,(y_1^{(0)},\ldots,y_n^{(0)})}, f^{(1)}_{n,\lambda,(y_1^{(1)},\ldots,y_n^{(1)})} \in \mathcal{F}}$ *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* $(i, \mathsf{tag\text{-}f}, y_i^{(0)}, y_i^{(1)}) \in \mathcal{Q}_{\mathsf{KGen}}$ *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*
   - $f^{(0)}_{n,\lambda,(y_1^{(0)},\ldots,y_n^{(0)})}(x_1^{(0)},\ldots,x_n^{(0)}) \neq f^{(1)}_{n,\lambda,(y_1^{(1)},\ldots,y_n^{(1)})}(x_1^{(1)},\ldots,x_n^{(1)})$ OR
     $f^{(0)}_{n,\lambda,(y_1^{(0)},\ldots,y_n^{(0)})}(x_1^{(0)},\ldots,x_n^{(0)}) \neq f^{(1)}_{n,\lambda,(y_1^{(1)},\ldots,y_n^{(1)})}(x_1^{(0)},\ldots,x_n^{(0)})$ OR
     $f^{(1)}(x_1^{(0)},\ldots,x_n^{(0)}) \neq f^{(1)}_{n,\lambda,(y_1^{(1)},\ldots,y_n^{(1)})}(x_1^{(1)},\ldots,x_n^{(1)})$.

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

### A.4 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 $\lambda$. 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 $\mathsf{ppt}$ adversary $\mathcal{A}$, the following advantage is negligible in $\lambda$:

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

where $K \xleftarrow{\$} \mathcal{K}$ and $R \xleftarrow{\$} \mathcal{R}$.
It is well-known that PRFs can be constructed under $\mathsf{DDH}$, *e.g.* the Naor-Reingold construction [48]. or under the *Learning with Rounding* (LWR) [16]. The LWR problem is shown to be as hard as $\mathsf{LWE}$ if the modulus and modulus-to-noise ratio[9] are super-polynomial [16, 11].

## A.5 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 $1^\lambda$, the algorithm outputs the public parameters $\mathsf{pp}$.
$\mathsf{KeyGen}(\mathsf{pp})$**:** On input the public parameters $\mathsf{pp}$, the algorithm outputs a pair $(\mathsf{sk}, \mathsf{pk})$ consisting of a secret key $\mathsf{sk}$ and the corresponding public key $\mathsf{pk}$.
$\mathsf{SharedKey}(\mathsf{sk}, \mathsf{pk}')$**:** On input a secret key $\mathsf{sk}$ and a (usually non-corresponding) public key $\mathsf{pk}'$, 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

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

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

**Security.** For a NIKE scheme $\mathcal{N}$ and a $\mathsf{ppt}$ 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 $\mathsf{pk}_1$ and $\mathsf{pk}_2$ 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 $\mathsf{pk}_1$ and $\mathsf{pk}_2$ 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 $\mathsf{ppt}$ adversary $\mathcal{A}$, the following advantage is negligible in $\lambda$:

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

NIKE can be constructed based on a variant of the Decisional Bilinear Diffie-Hellman assumption in the standard model [35, Section 4.3]. In a recent work [43], it is shown that NIKE can be constructed from $\mathsf{LWE}$ with polynomial modulus-to-noise ratio and satisfy strong security properties in the standard model.

[9] The modulus-to-noise ratio $q/\sigma$ is defined for the LWE problem over the ring of integers modulo $q$ and the errors are sampled from a discrete Gaussian distribution $D_\sigma$. The *approximation factor* in reducing LWE to a worst-case lattice problem relates closely to this ratio.

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

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

$\underline{\mathcal{O}\mathsf{Reveal}(\mathsf{pk}_1, \mathsf{pk}_2)}$:
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 $\{(\mathsf{sk}_1, \mathsf{pk}_1), (\mathsf{sk}_2, \mathsf{pk}_2)\} \nsubseteq \mathcal{H}$,
    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 $\mathsf{sk}$ s.t. $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{H}$
$\mathcal{H} \leftarrow \mathcal{H} \setminus \{(\mathsf{sk}, \mathsf{pk})\}$
Return $\mathsf{sk}$

$\underline{\mathsf{Finalize}(b')}$:
If $\mathsf{adm}(\mathcal{A}) = 1$, return $\beta \leftarrow (b' \overset{?}{=} b)$
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.6 From One-Challenge to Multi-Challenge – Proof of Lemma 5

**Lemma 5.** *Let* $\mathcal{E} = (\mathsf{GSetup}, \mathsf{LSetup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a DDFE scheme for a functionality* $f$. *If* $\mathcal{E}$ *is* one-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*

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

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

*Proof.* Let $\mathcal{A}$ be a ppt adversary in the experiment $\mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{A}}(1^\lambda)$ and $b \xleftarrow{\$} \{0, 1\}$ be the challenge bit. We denote the $q_e$ distinct $m_{\text{pub}}$ 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 $k_{\text{pub}}$ 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 $\mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{A}}(1^\lambda)$ 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 decryption key for $(k^{(1)}_{\text{pri}}, k^\ell_{\text{pub}})$ if $\ell \leq j$, and by a decryption key for $(k^{(0)}_{\text{pri}}, k^\ell_{\text{pub}})$ if $\ell > j$. Note that $\mathsf{G}_{1,0} = \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{A}}(1^\lambda)$, 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 22.

**Game $\mathsf{G}_{2,j}$ for $j \in [0; q_e]$:** This hybrid is the same as $\mathsf{G}_{1,q_k}$ 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 $\mathsf{G}_{2,q_e} = \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{A}}(1^\lambda)$, 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 23.

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:

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

where the last inequality is a consequence of Lemmas 22 and 23. □

**Lemma 22.** *If $\mathcal{E}$ is weakly function-hiding, then we have for each $j \in [q_k]$ that*

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

*Proof.* Let $\mathcal{A}$ be an adversary trying to distinguish between $\mathsf{G}_{1,j}$ and $\mathsf{G}_{1,j-1}$. We construct a $\mathsf{ppt}$ adversary $\mathcal{B}$ playing against $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ on input $(i, (m^0_{\mathrm{pri}}, m^\ell_{\mathrm{pub}}), (m^0_{\mathrm{pri}}, m^\ell_{\mathrm{pub}}))$ and forwards the response to $\mathcal{A}$.
- *Key-generation Queries:*
  Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ 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 $\mathcal{A}$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$. 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

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

and the lemma is concluded. □

**Lemma 23.** *If $\mathcal{E}$ is weakly function-hiding, then we have for each $j\in[q_e]$ that*

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

*Proof.* Let $\mathcal{A}$ be an adversary trying to distinguish between $\mathsf{G}_{2,j}$ and $\mathsf{G}_{2,j-1}$. We construct a $\mathsf{ppt}$ adversary $\mathcal{B}$ playing against $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.
- *Encryption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i,(m^{(0)}_{\text{pri}},m^\ell_{\text{pub}}),(m^{(1)}_{\text{pri}},m^\ell_{\text{pub}}))$, $\mathcal{B}$ behaves as follows:
  1. If $\ell<j$, $\mathcal{B}$ queries $(i,(m^{(1)}_{\text{pri}},m^\ell_{\text{pub}}),(m^{(1)}_{\text{pri}},m^\ell_{\text{pub}}))$ to the oracle $\mathcal{O}\mathsf{Enc}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.
  2. If $\ell=j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Enc}(i,(m^{(0)}_{\text{pri}},m^\ell_{\text{pub}}),(m^{(1)}_{\text{pri}},m^\ell_{\text{pub}}))$ and forwards the response to $\mathcal{A}$.
  3. If $\ell>j$, $\mathcal{B}$ queries $\mathcal{O}\mathsf{Enc}(i,(m^{(0)}_{\text{pri}},m^\ell_{\text{pub}}),(m^{(0)}_{\text{pri}},m^\ell_{\text{pub}}))$ and forwards the response to $\mathcal{A}$.
- *Key-generation Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{KeyGen}(i,(k^0_{\text{pri}},k^\ell_{\text{pub}}),(k^1_{\text{pri}},k^\ell_{\text{pub}}))$, $\mathcal{B}$ queries $(i,(k^{(1)}_{\text{pri}},k^\ell_{\text{pub}}),(k^{(1)}_{\text{pri}},k^\ell_{\text{pub}}))$ the oracle $\mathcal{O}\mathsf{Enc}$ of $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ and forwards the response to $\mathcal{A}$.
- *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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$ 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 $\mathcal{A}$ 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 $\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}wfh}}_{\mathcal{E},f,\mathcal{B}}(1^\lambda)$. 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 22, we conclude that

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

and the proof is completed. □

### A.7 From Weak to Full Function-Hiding – Proof of Lemma 6

**Lemma 6.** *If there exists a weakly function-hiding DDFE scheme $\mathcal{E}$ for $f^{\mathsf{dyn\text{-}ip}}$, then there exists a (fully) function-hiding DDFE scheme $\mathcal{E}'$ for $f^{\mathsf{dyn\text{-}ip}}$. More precisely, for any $\mathsf{ppt}$ adversary $\mathcal{A}$, there exists a $\mathsf{ppt}$ algorithm $\mathcal{B}$ such that*

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

*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 $f^{\mathsf{dyn\text{-}ip}}$ as follows:

- *Global Setup:* $\mathsf{GSetup}'(1^\lambda)$ runs $(\mathsf{pp}) \leftarrow \mathsf{GSetup}(1^\lambda)$ and outputs the public parameters $\mathsf{pp}$.
- *Local Setup:* $\mathsf{LSetup}'(\mathsf{pp})$ 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 $\mathsf{dk}_i \leftarrow \mathsf{KeyGen}(\mathsf{sk}_i, k_i = (\mathbf{y}_i \parallel 0^N, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))$ 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 $\mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, m_i = (\mathbf{x}_i \parallel 0^N, \mathcal{U}_{M,i}, \mathsf{tag}_i))$ 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

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

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 $\mathsf{G}_0$ equals $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}', f^{\mathsf{dyn\text{-}ip}}, \mathcal{A}}(1^\lambda)$, where the challenge bit is 0, and $\mathsf{G}_3$ equals $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}', f^{\mathsf{dyn\text{-}ip}}, \mathcal{A}}(1^\lambda)$, where the challenge bit is 1, and $\mathcal{A}$ is a $\mathsf{ppt}$ adversary. For $i \in [3]$, we denote the event $\mathsf{G}_i = 1$ to signify that $\mathcal{A}$ outputs 1 in the hybrid $\mathsf{G}_i$.

**Game $\mathsf{G}_0$:** This is $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}', f^{\mathsf{dyn\text{-}ip}}, \mathcal{A}}(1^\lambda)$ conditioned on the challenge bit $b = 0$. We recall that in this specific functionality $f^{\mathsf{dyn\text{-}ip}}$, there is the concept of tags in the public information of keys and ciphertexts. We denote the $\ell$-th distinct tag that occurs in a query to $\mathcal{O}\mathsf{Enc}$ by $\mathsf{tag}_\ell$. 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
$$\mathsf{ct}_{\ell,i} \leftarrow \mathsf{Enc}(i, (\mathbf{x}_i^{(0)} \parallel 0^N, \mathcal{U}_{M,i}, \mathsf{tag}_i), (\mathbf{x}_i^{(0)} \parallel 0^N, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
and returns $\mathsf{ct}'_{\ell,i} := \mathsf{ct}_{\ell,i}$.
- 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
$$\mathsf{dk}_{k,i} \leftarrow \mathsf{KeyGen}(i, (\mathbf{y}_i^{(0)} \parallel 0^N, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i), (\mathbf{y}_i^{(0)} \parallel 0^N, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i))$$
and returns $\mathsf{dk}'_{k,i} := \mathsf{dk}_{k,i}$.

We note that $N_i$ are included in $\mathsf{pk}_i$ 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 $(\mathbf{x}_i^{(0)} \parallel 0^N)_{i\in\mathcal{U}_M}$, 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 $(\mathbf{y}_i^{(0)} \parallel 0^N)_{i\in\mathcal{U}_K}$.

**Game $\mathsf{G}_1$:** 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
$$\mathsf{ct}_{\ell,i} \leftarrow \mathsf{Enc}(i, (0^N \parallel \mathbf{x}_i^{(1)}, \mathcal{U}_{M,i}, \mathsf{tag}_i), (0^N \parallel \mathbf{x}_i^{(1)}, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
and returns $\mathsf{ct}'_{\ell,i} := \mathsf{ct}_{\ell,i}$.

- 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 $(0^N \parallel \mathbf{x}_i^{(1)})_{i\in\mathcal{U}_M}$ (as opposed to $(\mathbf{x}_i^{(0)} \parallel 0^N)_{i\in\mathcal{U}_M}$ in $\mathsf{G}_0$), 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 $(\mathbf{y}_i^{(0)} \parallel 0^N)_{i\in\mathcal{U}_K}$ in $\mathsf{G}_0$). Let $n \coloneqq |\mathcal{U}_K| = |\mathcal{U}_M|$ in case of correct evaluation in $f^{\mathsf{dyn\text{-}ip}}$. 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
$$\begin{aligned}\left\langle \left(\mathbf{x}_i^{(0)} \parallel 0^N\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel 0^N\right)_{i\in[n]} \right\rangle &= \left\langle \left(\mathbf{x}_i^{(0)} \parallel 0^N\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle \\ &= \left\langle \left(0^N \parallel \mathbf{x}_i^{(1)}\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle\end{aligned}$$

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 $\mathsf{ppt}$ adversary $\mathcal{B}$ such that
$$\begin{aligned}|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_0 = 1]| &= \Big|\Pr\big[\mathcal{B} \text{ outputs 1 in } \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}, f^{\mathsf{dyn\text{-}ip}}, \mathcal{B}}(1^\lambda) \mid b = 1\big] \\ &\quad - \Pr\big[\mathcal{B} \text{ outputs 1 in } \mathbf{Exp}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}, f^{\mathsf{dyn\text{-}ip}}, \mathcal{B}}(1^\lambda) \mid b = 0\big]\Big| \\ &\le 2 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}, f^{\mathsf{dyn\text{-}ip}}, \mathcal{B}}(1^\lambda)\end{aligned}$$

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 $\mathsf{G}_2$:** 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
$$\mathsf{ct}_{\ell,i} \leftarrow \mathsf{Enc}(i, (\mathbf{x}_i^{(1)} \parallel 0^N, \mathcal{U}_{M,i}, \mathsf{tag}_i), (\mathbf{x}_i^{(1)} \parallel 0^N, \mathcal{U}_{M,i}, \mathsf{tag}_i))$$
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 $(\mathbf{x}_i^{(1)} \parallel 0^N)_{i\in\mathcal{U}_M}$ and a decryption key for $(\mathbf{y}_i^{(1)} \parallel \mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$, as opposed to $(0^N \parallel \mathbf{x}_i^{(1)})_{i\in\mathcal{U}_M}$ and $(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$ in $\mathsf{G}_1$. Let $n \coloneqq |\mathcal{U}_K| = |\mathcal{U}_M|$ in case of correct evaluation in $f^{\mathsf{dyn\text{-}ip}}$. Notice that
$$\begin{aligned}\left\langle \left(0^N \parallel \mathbf{x}_i^{(1)}\right)_{i\in[n]}, \left(\mathbf{y}_i^{(0)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle &= \left\langle \left(0^N \parallel \mathbf{x}_i^{(1)}\right)_{i\in[n]}, \left(\mathbf{y}_i^{(1)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle \\ &= \left\langle \left(\mathbf{x}_i^{(1)} \parallel 0^N\right)_{i\in[n]}, \left(\mathbf{y}_i^{(1)} \parallel \mathbf{y}_i^{(1)}\right)_{i\in[n]} \right\rangle .\end{aligned}$$

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 $\mathsf{ppt}$ adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1]| \le 2 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}, f^{\mathsf{dyn\text{-}ip}}, \mathcal{B}}(1^\lambda)$.

**Game $\mathsf{G}_3$:** We modify the definition of $\mathcal{O}\mathsf{KeyGen}$ as follows. (The definition of $\mathcal{O}\mathsf{Enc}$ is as in $\mathsf{G}_2$.)

- 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
$$\mathsf{dk}_{k,i} \leftarrow \mathsf{KeyGen}(i, (\mathbf{y}_i^{(1)} \parallel 0^N, \mathcal{U}_{K,i}, \mathsf{tag\text{-}f}_i), (\mathbf{y}_i^{(1)} \parallel 0^N, \mathsf{tag\text{-}f}_i))$$
and returns $\mathsf{dk}'_{k,i} := \mathsf{dk}_{k,i}$.

Thus, the challenger provides a decryption key for $(\mathbf{y}_i^{(1)} \parallel 0^N)_{i\in\mathcal{U}_K}$, as opposed to $(\mathbf{y}_i^{(1)} \parallel \mathbf{y}_i^{(1)})_{i\in\mathcal{U}_K}$ in $\mathsf{G}_2$. Let $n := |\mathcal{U}_K| = |\mathcal{U}_M|$ in case of correct evaluation in $f^{\mathsf{dyn\text{-}ip}}$. We have
$$\left\langle (\mathbf{x}_i^{(1)} \parallel 0^N)_{i\in[n]}, (\mathbf{y}_i^{(1)} \parallel \mathbf{y}_i^{(1)})_{i\in[n]} \right\rangle = \left\langle (\mathbf{x}_i^{(1)} \parallel 0^N)_{i\in[n]}, (\mathbf{y}_i^{(1)} \parallel 0^N)_{i\in[n]} \right\rangle .$$
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 $\mathsf{ppt}$ adversary $\mathcal{B}$ such that $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \le 2 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}, f^{\mathsf{dyn\text{-}ip}}, \mathcal{B}}(1^\lambda)$. Note that $\mathsf{G}_3$ equals the experiment $\mathbf{Exp}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}', f^{\mathsf{dyn\text{-}ip}}, \mathcal{A}}(1^\lambda)$ conditioned on $b = 1$.

Using a hybrid argument, we conclude that:
$$\mathbf{Adv}^{\mathsf{xxx\text{-}fh}}_{\mathcal{E}', f^{\mathsf{dyn\text{-}ip}}, \mathcal{A}}(1^\lambda) = \frac{1}{2}|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_0 = 1]| \le \frac{1}{2} \cdot \sum_{i=1}^{3} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i-1} = 1]|$$
$$\le 3 \cdot \mathbf{Adv}^{\mathsf{xxx\text{-}wfh}}_{\mathcal{E}, f^{\mathsf{dyn\text{-}ip}}, \mathcal{B}}(1^\lambda)$$
and the lemma is proved. □

## B Supporting Materials – Section 4

### B.1 Details about our DDFE in Section 4

**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
$$\sum_{i\in\mathcal{U}_K} \sum_{j\in\mathcal{U}_K\setminus\{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_K) = 0$$
Thus, $(s_i := \sum_{j\in\mathcal{U}_K\setminus\{i\}} (-1)^{j<i} F'_{K'_{i,j}}(\mathcal{U}_K))_{i\in\mathcal{U}_K} \in \mathcal{S}(|\mathcal{U}_K|, \mathbb{A})$. The argument for $\mathcal{U}_M$ proceeds in exactly the same way. Then the correctness of $\mathcal{E}$ follows from the correctness of $\mathcal{E}'$ and the decomposition of the setup algorithm according to the dynamizability.

**Security** We prove the security of our $\mathsf{DDFE}$ scheme below.

**Theorem 13.** *If $\mathcal{N}$ is an IND-secure NIKE scheme, $\{F_K\}_{K\in\mathcal{K}}$ and $\{F'_{K'}\}_{K'\in\mathcal{K}'}$ are families of pseudorandom functions and $\mathcal{E}'$ is a dynamizable (function-hiding)* $\mathsf{DMCFE}$ *scheme for a function class $\mathcal{F}$, then the* $\mathsf{DDFE}$ *scheme $\mathcal{E}$ in Fig. 2 for the functionality $f^{\mathsf{dyn}}$ corresponding to $\mathcal{F}$ is also (function-hiding) secure. 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* $\mathsf{ppt}$ *adversary $\mathcal{A}$, there exist* $\mathsf{ppt}$ *algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_4$ such that*
$$\mathbf{Adv}^{\mathsf{stat\text{-}xxx\text{-}cpa}}_{\mathcal{E}, f^{\mathsf{dyn}}, \mathcal{A}}(1^\lambda) \le q_h \cdot \mathbf{Adv}^{\mathsf{prf}}_{\{F_K\}, \mathcal{B}_1}(1^\lambda) + q_h^2 \cdot \mathbf{Adv}^{\mathsf{nike}}_{\mathcal{N}, \mathcal{B}_2}(1^\lambda)$$
$$+ q_h^2 \cdot \mathbf{Adv}^{\mathsf{prf}}_{\{F'_{K'}\}, \mathcal{B}_3}(1^\lambda) + q_u \cdot \mathbf{Adv}^{\mathsf{stat\text{-}xxx\text{-}cpa}}_{\mathcal{E}', \mathcal{F}, \mathcal{B}_4}(1^\lambda) ,$$
*where* $\mathsf{xxx} \subseteq \{\mathsf{1chal}, \mathsf{pos}, \mathsf{sel}, \mathsf{wfh}, \mathsf{fh}\}$.

*Proof.* We consider the case that $\mathsf{xxx} \cap \{\mathsf{wfh}, \mathsf{fh}\} \neq \varnothing$. Standard security (*i.e.* $\mathsf{xxx} \cap \{\mathsf{wfh}, \mathsf{fh}\} = \varnothing$) can then be treated as a special case where inputs to $\mathcal{O}\mathsf{KeyGen}(i, k_i^{(0)}, k_i^{(1)})$ are always of the form $(i, k_i, k_i)$. The proof is done via a sequence of hybrid games.

**Game $\mathsf{G}_0$:** This is the game $\mathbf{Exp}^{\mathsf{stat\text{-}xxx\text{-}cpa}}_{\mathcal{E}, f^{\mathsf{dyn}}, \mathcal{A}}(1^\lambda)$.

**Game $\mathsf{G}_1$:** 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 $\mathsf{G}_2$:** 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 $\mathsf{G}_3$:** 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 $\mathsf{G}_3$, for each set $\mathcal{U}$ that occurs in an encryption or key-generation query, $(s_i := \sum_{j \in \mathcal{U} \setminus \{i\}} (-1)^{j<i} R'_{i,j}(\mathcal{U}_K))_{i \in \mathcal{U} \cap \mathcal{H}}$ is uniformly random subject to the condition that $\sum_{i \in \mathcal{U} \cap \mathcal{H}} s_i = -\sum_{i \in \mathcal{U} \cap \mathcal{C}} s_i$. Thus, by the dynamizability of $\mathcal{E}'$, it follows for $i \in \mathcal{U} \cap \mathcal{H}$ that $\mathsf{sk}'_i := P_{\mathsf{sk}}(s_i, \widehat{\mathsf{sk}}_i)$ and $\mathsf{ek}'_i := P_{\mathsf{ek}}(s_i, \widehat{\mathsf{ek}}_i)$ computed for responses to $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Enc}$ queries now follow the same distribution as "real" keys generated by $\mathcal{E}'.\mathsf{Setup}$. This allows us to rely on the security of $\mathcal{E}'$ in what follows.

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 $\mathsf{G}_3$ 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

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

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

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

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 $|\Pr[\hat{\mathsf{G}}_\kappa = 1] - \Pr[\hat{\mathsf{G}}_{\kappa-1} = 1]| \leq \mathbf{Adv}^{\mathsf{xxx\text{-}stat\text{-}fh}}_{\mathcal{E}', \mathcal{F}, \mathcal{B}_4}(1^\lambda)$. □

### B.2 From Complete to Incomplete Challenges – Proof of Lemma 14

**Lemma 14.** *Assume there exist (1) a one-challenge (weakly function-hiding) DDFE scheme $\mathcal{E}^{\mathsf{pos}}$ for a functionality $f^{\mathsf{dyn}}$ 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 $f^{\mathsf{dyn}}$ that is even secure against incomplete queries. More precisely, for any* ppt *adversary $\mathcal{A}$, there exist* ppt *algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ such that*

$$\mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}yyy\text{-}cpa}}_{\mathcal{E}, f^{\mathsf{dyn}}, \mathcal{A}}(1^\lambda) \leq 6 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}xxx\text{-}yyy\text{-}cpa}}_{\mathcal{E}^{\mathsf{pos}}, f^{\mathsf{dyn}}, \mathcal{B}_1}(1^\lambda) + 6 \cdot \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}cpa}}_{\mathcal{E}^{\mathsf{aone}}, f^{\mathsf{aone}}, \mathcal{B}_2}(1^\lambda) \ ,$$

*where* $\mathsf{xxx} \subseteq \{\mathsf{stat}, \mathsf{sel}\}$ *and* $\mathsf{yyy} \subseteq \{\mathsf{wfh}\}$.

*Proof.* We consider only the weakly function-hiding setting, *i.e.* $\mathsf{yyy} = \{\mathsf{wfh}\}$. Standard security (*i.e.* $\mathsf{yyy} = \varnothing$) can then be treated as a special case where inputs to $\mathcal{O}\mathsf{KeyGen}(i, k_i^{(0)}, k_i^{(1)})$ are always of the form $(i, k_i, k_i)$.
Let $\mathcal{E}^{\mathsf{pos}} = (\mathsf{pGSetup}, \mathsf{pLSetup}, \mathsf{pKeyGen}, \mathsf{pEnc}, \mathsf{pDec})$ be a one-challenge, weakly function-hiding $\mathsf{DDFE}$ scheme for the function class $f^{\mathsf{dyn}}$ that is secure against complete queries, and let $\mathcal{E}^{\mathsf{aone}} = (\mathsf{aGSetup}, \mathsf{aLSetup}, \mathsf{aEnc}, \mathsf{aDec})$ be a $\mathsf{DDFE}$ scheme for the $\mathsf{AoNE}$ functionality $\mathcal{F}^{\mathsf{aone}}$. We construct a one-challenge, weakly function-hiding $\mathsf{DDFE}$ scheme $\mathcal{E}$ for the function class $f^{\mathsf{dyn}}$ 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 $1^\lambda$, 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 $\mathsf{PP}$ 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 $\mathsf{SK}_i$ and $k = (k_{\mathrm{pri}}, k_{\mathrm{pub}})$, compute

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

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

$\mathsf{Enc}(\mathsf{SK}_i, m)$**:** On input a secret key $\mathsf{SK}_i$ 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 $\mathsf{CT}_i := \mathsf{aCT}_i$.

$\mathsf{Dec}((\mathsf{DK}_i)_{i \in \mathcal{U}_K}, (\mathsf{CT}_i)_{i \in \mathcal{U}_M})$**:** 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 $\mathbf{Adv}^{\mathsf{G}_i}(\mathcal{A}) := |\Pr[\mathsf{G}_i = 1] - 1/2|$. To improve readability, we introduce the shorthands

$$\left(\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}, \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}_1}\right) := \left(\mathbf{Exp}^{\mathsf{1chal\text{-}pos\text{-}xxx\text{-}yyy\text{-}cpa}}_{\mathcal{E}^{\mathsf{pos}}, f^{\mathsf{dyn}}, \mathcal{B}_1}(1^\lambda), \mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}xxx\text{-}yyy\text{-}cpa}}_{\mathcal{E}^{\mathsf{pos}}, f^{\mathsf{dyn}}, \mathcal{B}_1}(1^\lambda)\right)$$
$$\left(\mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}_2}, \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}_2}\right) := \left(\mathbf{Exp}^{\mathsf{1chal\text{-}xxx\text{-}cpa}}_{\mathcal{E}^{\mathsf{aone}}, f^{\mathsf{aone}}, \mathcal{B}_2}(1^\lambda), \mathbf{Adv}^{\mathsf{1chal\text{-}xxx\text{-}cpa}}_{\mathcal{E}^{\mathsf{aone}}, \mathcal{F}^{\mathsf{aone}}, \mathcal{B}_2}(1^\lambda)\right) \ .$$

The hybrid games are defined as follows.

**Game $\mathsf{G}_0$:** This game equals $\mathbf{Exp}^{\text{1chal-xxx-yyy-cpa}}_{\mathcal{E},f^{\text{dyn}},\mathcal{A}}(1^\lambda)$, so we have $\mathbf{Adv}^{\mathsf{G}_0} = \mathbf{Adv}^{\text{1chal-xxx-yyy-cpa}}_{\mathcal{E},f^{\text{dyn}},\mathcal{A}}(1^\lambda)$. 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 $m^*_{\text{pub}} = (\mathcal{U}^*_M, \mathsf{tag}^*), k^*_{\text{pub}} = (\mathcal{U}^*_K, \mathsf{tag\text{-}f}^*)$ so that:

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

We define events $E_0$ and $E_1$ as follows:

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

**Game $\mathsf{G}_1$:** This is the same as $\mathsf{G}_0$ 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 0 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 $\mathsf{G}_2$:** If $d = 1$, then the simulation works exactly as in the previous game. Otherwise, the simulator acts as an adversary $\mathcal{B}_1$ in the game $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}$. Specifically, if $d = 0$, the simulation works as follows. W.l.o.g., we assume that each $i \in \mathsf{ID}$ is queried at most once to $\mathcal{O}\mathsf{HonestGen}$ and $\mathcal{O}\mathsf{Corrupt}$.

- *Initialization:* Upon $\mathcal{A}$ calling $\mathsf{Initialize}(1^\lambda, m^*_{\text{pub}}, k^*_{\text{pub}})$, $\mathcal{B}_1$ chooses a random bit $b \xleftarrow{\$} \{0,1\}$, initializes empty sets $\mathcal{C}$ and $\mathcal{H}$, runs
$$\mathsf{pPP} \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathsf{Initialize}(1^\lambda); \quad \mathsf{aPP} \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{GSetup}(1^\lambda)$$
and returns $\mathsf{PP} := (\mathsf{pPP}, \mathsf{aPP})$.
- *User-Generation Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{HonestGen}(i)$ for some $i \in \mathsf{ID}$, $\mathcal{B}_1$ queries and computes
$$\mathsf{pPK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathcal{O}\mathsf{HonestGen}(i); \quad (\mathsf{aSK}_i, \mathsf{aPK}_i) \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{LSetup}(i) \ ,$$
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 $i \in \mathsf{ID}$, $\mathcal{B}_1$ first checks whether $\mathcal{O}\mathsf{Corrupt}$ has previously been called on the same input and returns the same calue as before in this case. Otherwise, $\mathcal{B}_1$ checks whether $i \in \mathcal{H}$ and calls $\mathcal{O}\mathsf{HonestGen}(i)$ if this is not the case yet. It then removes $i$ from $\mathcal{H}$, adds it to $\mathcal{C}$, queries $\mathsf{pSK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathcal{O}\mathsf{Corrupt}(i)$ and returns $\mathsf{SK}_i := (\mathsf{pSK}_i, \mathsf{aSK}_i)$, where $\mathsf{aSK}_i$ is known from the corresponding query to $\mathcal{O}\mathsf{HonestGen}$.
- *Encryption Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{Enc}(i, (m^{(0)}_{\text{pri}}, m_{\text{pub}}), (m^{(1)}_{\text{pri}}, m_{\text{pub}}))$, $\mathcal{B}_1$ queries and computes
$$\mathsf{pCT}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathcal{O}\mathsf{Enc}(i, (m^{(b)}_{\text{pri}}, m_{\text{pub}}), (m^{b}_{\text{pri}}, m_{\text{pub}})); \quad \mathsf{aCT}_i \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{Enc}(\mathsf{aSK}_i, (\mathsf{pCT}_i, m_{\text{pub}}))$$
and returns $\mathsf{CT}_i := \mathsf{aCT}_i$.
- *Key-Generation Queries:* Upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}$ on input $(i, (k^{(0)}_{\text{pri}}, k_{\text{pub}}), (k^{(1)}_{\text{pri}}, k_{\text{pub}}))$, $\mathcal{B}_1$ does the following:
  - If $k_{\text{pub}} = k^*_{\text{pub}}$, $\mathcal{B}_1$ queries
$$\mathsf{pDK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathcal{O}\mathsf{KeyGen}(i, (k^{(b)}_{\text{pri}}, k^*_{\text{pub}}), (k^{(1)}_{\text{pri}}, k^*_{\text{pub}})); \quad \mathsf{aDK}_i \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{Enc}(\mathsf{aSK}_i, (\mathsf{pDK}_i, k^*_{\text{pub}}))$$
and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.

  - ∘ If $k_{\text{pub}} \neq k^*_{\text{pub}}$, then $k^{(0)}_{\text{pri}} = k^{(1)}_{\text{pri}}$ and $\mathcal{B}_1$ queries

    $$\mathsf{pDK}_i \leftarrow \mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathcal{O}\mathsf{KeyGen}(i, (k^{(1)}_{\text{pri}}, k^*_{\text{pub}}), (k^{(1)}_{\text{pri}}, k^*_{\text{pub}})); \quad \mathsf{aDK}_i \leftarrow \mathcal{E}^{\mathsf{aone}}.\mathsf{Enc}(\mathsf{aSK}_i, (\mathsf{pDK}_i, k^*_{\text{pub}}))$$

    and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.
- *Finalize:* Upon $\mathcal{A}$ calling $\mathsf{Finalize}(b')$, $\mathcal{B}_1$ forwards the same bit to its own challenger by calling $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}.\mathsf{Finalize}(b')$.

In the end, we have $|\mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{G}_1}(\mathcal{A})| \leq \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}_1}$.
**Game $\mathsf{G}_3$:** We do a similar modification in the simulation for the case $d = 1$. That is, for queries of the form $\mathcal{O}\mathsf{KeyGen}((i, (k^{(0)}_{\text{pri}}, k^*_{\text{pub}}), (k^{(1)}_{\text{pri}}, k^*_{\text{pub}})))$, the simulator now outputs a key for $(k^{(1)}_{\text{pri}}, k^*_{\text{pub}})$ instead of $(k^{(b)}_{\text{pri}}, k^*_{\text{pub}})$. The indistinguishability between $\mathsf{G}_3$ and $\mathsf{G}_2$ reduces to the security of $\mathcal{E}^{\mathsf{aone}}$. We construct a reduction $\mathcal{B}_2$ that acts as an adversary in the experiment $\mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}_2}$. $\mathcal{B}_2$ replaces all $\mathcal{E}^{\mathsf{aone}}$ algorithms with calls to the respective oracles of $\mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}_2}$, in the same vein as $\mathcal{B}_1$ does with calls to oracles of $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}_1}$ in $\mathsf{G}_2$. In particular, upon $\mathcal{A}$ querying $\mathcal{O}\mathsf{DKeyGen}$ on input $(i, (k^{(0)}_{\text{pri}}, k_{\text{pub}}), (k^{(1)}_{\text{pri}}, k_{\text{pub}}))$, $\mathcal{B}_2$ does the following:

- If $k_{\text{pub}} = k^*_{\text{pub}}$, $\mathcal{B}_2$ queries

  $$\begin{aligned}
  &\mathsf{pDK}_i \leftarrow \mathcal{E}^{\mathsf{pos}}.\mathsf{KeyGen}(\mathsf{pSK}_i, (k^{(b)}_{\text{pri}}, k^*_{\text{pub}}))\\
  &\mathsf{pDK}'_i \leftarrow \mathcal{E}^{\mathsf{pos}}.\mathsf{KeyGen}(\mathsf{pSK}_i, (k^{(1)}_{\text{pri}}, k^*_{\text{pub}}))\\
  &\mathsf{aDK}_i \leftarrow \mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}_2}.\mathcal{O}\mathsf{Enc}(i, (\mathsf{pDK}_i, k^*_{\text{pub}}), (\mathsf{pDK}'_i, k^*_{\text{pub}}))
  \end{aligned}$$

  and returns $\mathsf{DK}_i := \mathsf{aDK}_i$.
- If $k_{\text{pub}} \neq k^*_{\text{pub}}$, then $k^{(0)}_{\text{pri}} = k^{(1)}_{\text{pri}}$ and $\mathcal{B}_2$ queries

  $$\begin{aligned}
  &\mathsf{pDK}_i \leftarrow \mathcal{E}^{\mathsf{pos}}.\mathsf{KeyGen}(\mathsf{pSK}_i, (k^{(1)}_{\text{pri}}, k_{\text{pub}}))\\
  &\mathsf{aDK}_i \leftarrow \mathbf{Exp}^{\mathsf{aone}}_{\mathcal{B}_2}.\mathcal{O}\mathsf{Enc}(i, (\mathsf{pDK}_i, k_{\text{pub}}), (\mathsf{pDK}_i, k_{\text{pub}}))
  \end{aligned}$$

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

In the end, we have $|\mathbf{Adv}^{\mathsf{G}_3}(\mathcal{A}) - \mathbf{Adv}^{\mathsf{G}_2}(\mathcal{A})| \leq \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}_2}$.
**Game $\mathsf{G}_4$:** We answer queries of the form $\mathcal{O}\mathsf{Enc}((i, (m^{(0)}_{\text{pri}}, m^*_{\text{pub}}), (m^{(1)}_{\text{pri}}, m^*_{\text{pub}})))$ by encryptions of $(m^{(1)}_{\text{pri}}, m^*_{\text{pub}})$ as opposed to $(m^{(b)}_{\text{pri}}, m^*_{\text{pub}})$ using a similar sequence of hybrids as $\mathsf{G}_1$, $\mathsf{G}_2$ and $\mathsf{G}_3$, but with flipped roles of the oracles $\mathcal{O}\mathsf{KeyGen}$ and $\mathcal{O}\mathsf{Enc}$. Note that $\mathsf{G}_4$ is independent of the bit $b$. In the end, we obtain $\mathbf{Adv}^{\mathsf{G}_3}(\mathcal{A}) \leq 2 \cdot (\mathbf{Adv}^{\mathsf{G}_4}(\mathcal{A}) + \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}_1} + \mathbf{Adv}^{\mathsf{aone}}_{\mathcal{B}_2})$
To conclude, we compute

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

where the last equality follows from the fact that $\mathsf{G}_4$ is independent of $b$. □

## B.3 Instantiation of the Generic Conversion with Our $\mathsf{FH\text{-}DMCFE}$

The following lemma argues that our $\mathsf{DMCFE}$ depicted in Fig. 7 fits into the framework of dynamizable $\mathsf{DMCFE}$ schemes.

**Lemma 24.** *The IP-DMCFE in Fig. 7 is $\mathbb{Z}_q^2$-dynamizable.*

*Proof.* The scheme admits the following implementation of the algorithms $\mathsf{SetupPP}$ and $\mathsf{SetupUser}$.

$\mathsf{SetupPP}(1^\lambda)$**:** Run $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathrm{t}, g_1, g_2, g_\mathrm{t}, \mathbf{e}, q) \leftarrow \mathsf{GGen}(1^\lambda)$ and sample two full-domain hash functions $\mathsf{H}_1$ and $\mathsf{H}_2$ onto $\mathbb{G}_1^2$ and $\mathbb{G}_2^2$ respectively. Return $\widehat{\mathsf{pp}} := (\mathbb{G}, \mathsf{H}_1, \mathsf{H}_2)$.
$\mathsf{SetupUser}(\widehat{\mathsf{pp}})$**:** Generate $(B_i, B_i^*) \leftarrow \mathsf{DPVSGen}(\mathbb{G}, 1^{4N+5})$ and return $(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i)$ computed as follows:

$$\widehat{\mathsf{sk}}_i := (\mathbf{b}_{i,1}^*, \ldots, \mathbf{b}_{i,N}^*,\ B_{i,N+1}^*,\ B_{i,N+2}^*,\ \mathbf{b}_{i,N+3}^*)$$
$$\widehat{\mathsf{ek}}_i := (\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,N},\ B_{i,N+1},\ B_{i,N+2},\ \mathbf{b}_{i,N+4})$$

Furthermore, we define $P_{\mathsf{sk}}(\mathbf{s}_i = (\tilde{s}_i, \tilde{t}_i), \widehat{\mathsf{sk}}_i) := (\tilde{s}_i, \widehat{\mathsf{sk}}_i)$ and $P_{\mathsf{ek}}(\mathbf{s}_i = (\tilde{s}_i, \tilde{t}_i), \widehat{\mathsf{ek}}_i) := (\tilde{t}_i, \widehat{\mathsf{ek}}_i)$. We note that when $(\tilde{s}_1, \tilde{t}_1), \ldots, (\tilde{s}_n, \tilde{t}_n) \xleftarrow{\$} \mathbb{Z}_q^2$ satisfying $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$, $P_{\mathsf{sk}}(\mathbf{s}_i = (\tilde{s}_i, \tilde{t}_i), \widehat{\mathsf{sk}}_i)$ and $P_{\mathsf{ek}}(\mathbf{s}_i = (\tilde{s}_i, \tilde{t}_i), \widehat{\mathsf{ek}}_i) := (\tilde{t}_i, \widehat{\mathsf{ek}}_i)$ give the keys $\mathsf{sk}_i$ and $\mathsf{ek}_i$ in our construction of Fig. 7, respectively. Then the following distributions are equal

$$\left\{\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}\right\} = \left\{\widehat{\mathsf{pp}}, \left(P_{\mathsf{sk}}(\mathbf{s}_i, \widehat{\mathsf{sk}}_i), P_{\mathsf{ek}}(\mathbf{s}_i, \widehat{\mathsf{ek}}_i)\right)_{i\in[n]}\right\} \quad ,$$

where $(\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathbf{s}_1, \ldots, \mathbf{s}_n \xleftarrow{\$} \mathbb{Z}_q^2$ conditioned on $\sum_{i=1}^n \mathbf{s}_i = \mathbf{0}$. ◻

The $\mathsf{DMCFE}$ scheme in Fig. 7 was proven to be one-challenge, weakly function-hiding secure against complete queries under static corruption in the ROM. Our conversion yields a (fully) function-hiding $\mathsf{DDFE}$ scheme without the one-challenge and complete-queries constraints. Specifically, we obtain the following corollary of Corollary 15.

**Corollary 25.** *There exists a FH-DDFE scheme for the function class $f^{\mathsf{dyn\text{-}ip}}$ that is secure against static corruption under the $\mathsf{SXDH}$ assumption in the ROM.*

## B.4 Instantiation of the Generic Conversion with the $\mathsf{DMCFE}$ of [27]

We recall the construction of [27] in Fig. 5 using our notations. The scheme considers a restricted variant of the inner-product functionality $\mathcal{F}^{\mathsf{ip}}$ where each user encrypts sub-vectors of length 1. The original syntax used in [27] differs from ours in a few aspects, which we list below. For details, see Sections 2.3 and A.3.

- The scheme of [27] considers an additional algorithm $\mathsf{DKeyComb}$ that, given $n$ functional key components $(\mathsf{dk}_{\mathsf{tag\text{-}f},i})_{i\in[n]}$ generated for the same tag $\mathsf{tag\text{-}f}$, outputs a succinct functional key $\mathsf{dk}_{\mathsf{tag\text{-}f}}$ which can be passed to the decryption algorithm. Instead, we implicitly include the $\mathsf{DKeyComb}$ in the decryption algorithm.
- The key generation algorithm of [27] (called $\mathsf{DKeyGenShare}$ in their syntax) takes only two arguments, a secret key $\mathsf{sk}_i$ and a tag $\mathsf{tag}$ containing a description of the corresponding function. However, this syntax does not allow considering repetitions in the key generation. Therefore, our tags do not define the entire function, but we pass a third argument to the key generation algorithm containing the part of the function description necessary to compute the decryption key.

$\mathsf{Setup}(1^\lambda, 1^n)$**:** On input the security parameter $1^\lambda$ and the number of clients $1^n$, run $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q) \leftarrow \mathsf{GGen}(1^\lambda)$ and sample two full-domain hash functions $\mathsf{H}_1$ and $\mathsf{H}_2$ onto $\mathbb{G}_1^2$ and $\mathbb{G}_2^2$ respectively. For each $i \in [n]$, generate $\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_q^2$ and $\mathbf{T}_i \xleftarrow{\$} \mathbb{Z}_q^{2\times 2}$ such that $\sum_{i\in[n]} \mathbf{T}_i = 0$. Return $(\mathsf{pp} := (\mathbb{G}, \mathsf{H}_1, \mathsf{H}_2), (\mathsf{sk}_i := (\mathbf{s}_i, \mathbf{T}_i), \mathsf{ek}_i := \mathbf{s}_i)_{i\in[n]})$.

$\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag}\text{-}\mathsf{f}, y_i)$**:** On input a secret key $\mathsf{sk}_i = (\mathbf{s}_i, \mathbf{T}_i)$, a function tag $\mathsf{tag}\text{-}\mathsf{f}$ and a scalar $\mathbf{y}_i \in \mathbb{Z}_q$, compute $[\![\boldsymbol{\mu}]\!]_2 = \mathsf{H}_2(\mathsf{tag}\text{-}\mathsf{f})$, $[\![\mathbf{d}_i]\!]_2 = [\![y_i \cdot \mathbf{s}_i + \mathbf{T}_i \cdot \boldsymbol{\mu}]\!]_2$ and return $\mathsf{dk}_i := [\![\mathbf{d}_i]\!]_2$.

$\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, x_i)$**:** On input an encryption key $\mathsf{ek}_i = \mathbf{s}_i$, a tag $\mathsf{tag}$ and a scalar $x_i \in \mathbb{Z}_q$, compute $[\![\boldsymbol{\omega}]\!]_1 = \mathsf{H}_1(\mathsf{tag})$, $[\![c_i]\!]_1 = [\![\langle \boldsymbol{\omega}, \mathbf{s}_i\rangle + x_i]\!]_1$ and return $\mathsf{ct}_i := ([\![c_i]\!]_1, \mathsf{tag})$.

$\mathsf{Dec}((\mathsf{dk}_i)_{i\in[n]}, (\mathsf{ct}_i)_{i\in[n]})$**:** On input a list of decryption keys $(\mathsf{dk}_i := [\![\mathbf{d}_i]\!]_2)_{i\in[n]}$ and a list of ciphertexts $(\mathsf{ct}_i := ([\![c_i]\!]_1, \mathsf{tag}_i))_{i\in[n]}$, if $\mathsf{H}_1(\mathsf{tag}_1) = \cdots = \mathsf{H}_1(\mathsf{tag}_n) =: [\![\boldsymbol{\omega}]\!]_1$ compute

$$[\![\mathsf{out}]\!]_t = \textstyle\sum_{i=1}^n [\![c_i]\!]_1 \times [\![y_i]\!]_2 - [\![\boldsymbol{\omega}]\!]_1 \times [\![\mathbf{d}_i]\!]_2 \quad ,$$

then find and output the discrete log $\mathsf{out}$. Otherwise, abort with failure.

Fig. 5: DMCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ presented in [27]

- The decryption algorithm of [27] explicitly takes the ciphertext's tag $\mathsf{tag}$ as an additional argument. Without loss of generality, this tag can be included in the ciphertexts to match our syntax.

The following lemma shows that the construction of [27] fits into our framework of dynamizable DMCFE schemes.

**Lemma 26.** *The DMCFE scheme $\mathcal{E}$ in Fig. 5 is $\mathbb{Z}_q^{2\times 2}$-dynamizable.*

*Proof.* The scheme admits the following implementation of the algorithms SetupPP and SetupUser.

$\mathsf{SetupPP}(1^\lambda)$**:** Run $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, g_1, g_2, g_t, \mathbf{e}, q) \leftarrow \mathsf{GGen}(1^\lambda)$ and sample two full-domain hash functions $\mathsf{H}_1$ and $\mathsf{H}_2$ onto $\mathbb{G}_1^2$ and $\mathbb{G}_2^2$ respectively. Return $\widehat{\mathsf{pp}} := (\mathbb{G}, \mathsf{H}_1, \mathsf{H}_2)$.

$\mathsf{SetupUser}(\widehat{\mathsf{pp}})$**:** Sample $\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_q^2$ and return $(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i) := (\mathbf{s}_i, \mathbf{s}_i)$.

Furthermore, we define $P_{\mathsf{sk}}(\mathbf{T}_i, \widehat{\mathsf{sk}}_i) = (\mathbf{T}_i, \widehat{\mathsf{sk}}_i)$ to be the identity function and $P_{\mathsf{ek}}(\mathbf{T}_i, \widehat{\mathsf{ek}}_i) = \widehat{\mathsf{ek}}_i$ to be the projection onto the second coordinate. Then the following distributions are equal

$$\left\{\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}\right\} = \left\{\widehat{\mathsf{pp}}, \left(P_{\mathsf{sk}}(\mathbf{T}_i, \widehat{\mathsf{sk}}_i), P_{\mathsf{ek}}(\mathbf{T}_i, \widehat{\mathsf{ek}}_i)\right)_{i\in[n]}\right\} \quad ,$$

where $(\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i) \leftarrow \mathsf{SetupUser}(\widehat{\mathsf{pp}})$ for all $i \in [n]$, and $\mathbf{T}_1, \ldots, \mathbf{T}_n \xleftarrow{\$} \mathbb{Z}_q^{2\times 2}$ conditioned on $\sum_{i=1}^n \mathbf{T}_i = \mathbf{0}$. □

The IP-DMCFE scheme of [27] was proven to be adaptively secure without repetitions against complete queries under static corruption in the ROM. By an application of Corollary 15, we obtain the following result.

**Corollary 27.** *There exists an IP-DDFE scheme where each user encrypts vectors of length 1 that is adaptively secure without repetitions (neither for $\mathcal{O}$Enc nor for $\mathcal{O}$KeyGen queries) against static corruption under the SXDH assumption in the ROM.*

## B.5 Instantiation to Obtain an Adaptively Secure LWE-based DDFE for Inner Products

We recall the construction of [45] in Fig. 6. For the ease of comparison with [45], we now use $\ell$ to denote the number of clients, as well as the number of senders. The scheme considers a restricted variant of the inner-product functionality $\widetilde{\mathcal{F}^{\mathsf{ip}}} = \{\widetilde{\mathcal{F}^{\mathsf{ip}}}_{\ell,\lambda}\}_{\ell,\lambda}$ (compared to $\mathcal{F}^{\mathsf{ip}}$ from Definition 8) where $\widetilde{\mathcal{F}^{\mathsf{ip}}}_{\ell,\lambda} = \{\tilde{f}_{\ell,\lambda,(y_1,\ldots,y_\ell)} \colon \mathcal{D}_\lambda^\ell \to \mathcal{R}_\lambda\}_{(y_1,\ldots,y_\ell)\in\mathsf{Param}_\lambda^\ell}$ as the family of functions

$$\tilde{f}_{\ell,\lambda,(y_1,\ldots,y_\ell)}(\mathbf{x}_1,\ldots,\mathbf{x}_\ell) = \sum_{i=1}^{\ell} y_i \cdot \mathbf{x}_i$$

where $\mathcal{D}_\lambda \coloneqq [-X,X]^{n_0}$, $\mathsf{Param}_\lambda \coloneqq [-Y,Y]$ for some bound $\|\mathbf{y}\|_\infty \leq Y$ with $\mathbf{y} \coloneqq (y_1,\ldots,y_\ell)$. We then propose a variant of [45], where the differing details in $\boxed{\text{boxes}}$. A proof of security for our proposal is given in Theorem 30 and more importantly, this variant fits into our framework from Section 4 following Lemma 29.

**Preliminaries.** If $X$ and $Y$ are distributions over the same domain $\mathcal{D}$, then $\Delta(X,Y)$ denotes their statistical distance. For the preliminaries on lattices, homomorphic encryption, and admissible hash functions, we refer to [45, Section 2].

*Probability.* Let $\mathbf{\Sigma} \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and $\mathbf{c} \in \mathbb{R}^\ell$ be a vector. We define the *Gaussian function* over $\mathbb{R}^n$ by $\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x}) = \exp(-\pi(\mathbf{x}-\mathbf{c})^\top\mathbf{\Sigma}^{-1}(\mathbf{x}-\mathbf{c}))$ and if $\mathbf{\Sigma} = \sigma^2 \cdot \mathbf{I}_n$ and $\mathbf{c} = \mathbf{0}$, we write $\rho_\sigma$ for $\rho_{\mathbf{\Sigma},\mathbf{c}}$. For any discrete set $\Lambda \subset \mathbb{R}^n$, the *discrete Gaussian distribution* $D_{\Lambda,\mathbf{\Sigma},\mathbf{c}}$ has probability mass $\Pr_{X\sim D_{\Lambda,\mathbf{\Sigma},\mathbf{c}}}[X = \mathbf{x}] = \frac{\rho_{\mathbf{\Sigma},\mathbf{c}}(\mathbf{x})}{\rho_{\mathbf{\Sigma},\mathbf{c}}(\Lambda)}$, for any $\mathbf{x} \in \Lambda$. When $\mathbf{c} = \mathbf{0}$ and $\mathbf{\Sigma} = \sigma^2 \cdot \mathbf{I}_n$ we denote $D_{\Lambda,\mathbf{\Sigma},\mathbf{c}}$ by $D_{\Lambda,\sigma}$. We also apply the Chernoff-Cramér method to derive a tail bound for Gaussian random variables. The proof is classic and omitted.

**Theorem 28.** *Let $X \sim N(0,\nu)$ where $\nu > 0$ is the variance. For any $\beta > 0$, it holds that*

$$\Pr[|X| \geq \beta] \leq 2 \cdot \exp\left(-\frac{\beta^2}{2\nu}\right) \quad .$$

Finally, we also need some inequalities from calculus:

$$e^x \leq \left(1 + \frac{x}{n}\right)^{n+x/2} \quad ; \quad 1 + x \leq e^x \tag{10}$$

for $x, n > 0$.

**Constructions.** The construction of [45] and our variant $\boxed{\mathcal{E}}$ are presented in Fig. 6. We use the following global public parameters

$$\begin{aligned}\mathsf{cp} = (&\lambda, \ell_{\max}, X, \bar{X}, Y, \bar{Y}, n_0, n_1, \bar{n}_1, n_{\mathrm{g}}, \bar{n}_{\mathrm{g}}, m, \bar{m}, \alpha,\\ &\alpha_1, \bar{\alpha}_1, \sigma, \bar{\sigma}, \ell_t, \ell_f, L, q, \bar{q}, \mathsf{AHF}, \mathsf{AHF}_f, \boxed{\mathrm{M}})\end{aligned}$$

where $\boxed{\mathrm{M}}$ is a new additional parameter for our variant $\boxed{\mathcal{E}}$. The security parameter is $\lambda$ and other quantities follow the below setting:

- Let $\ell_{\max} = \lambda^k, n_1 = \lambda^d, \bar{d} = 3d+k-1, q = 2^{\lambda^{d-1}+\lambda}, q = 2^{\lambda^{\bar{d}-1}+\lambda}, \bar{n}_1 = \lambda^{\bar{d}}, \alpha_1 = 2^{-\lambda^{d-1}+d\log\lambda}, \bar{\alpha}_1 = 2^{-\lambda^{\bar{d}-1}+\bar{d}\log\lambda}$, $\alpha = 2^{-\sqrt{\lambda}}, n_0 \cdot \ell_{\max} = O(\lambda^{d-2})$, $n_0 = O(\lambda^{d-2}), n_{\mathrm{g}} = O(\lambda^{2d-1}), \bar{n}_{\mathrm{g}} = O(\lambda^{4d+k-2})$, $X = \bar{Y} = 1$, $\sigma = 2^{\lambda^{d-1}-2\lambda}, \bar{\sigma} = 2^{\lambda^{\bar{d}-1}-2\lambda}$, $\bar{X} = 2\ell \cdot Y \cdot \sigma\sqrt{n_{\mathrm{g}}}$ and the rest $Y, m, \bar{m} = \mathrm{poly}(\lambda)$.
- The tag lengths $\ell_t \in \Theta(\lambda)$ for encryption and $\ell_f \in \Theta(\lambda)$ for key generation.
- The dimensions $n_{\mathrm{g}}, m, n_0, n_1, \bar{n}_{\mathrm{g}}, \bar{m} \in \mathrm{poly}(\lambda)$ satisfy that $n_{\mathrm{g}} > 3 \cdot (n_0 + n_1) \cdot \lceil \log q \rceil$, $m > 2 \cdot n_{\mathrm{g}} \cdot \lceil \log q \rceil$, $\bar{n}_{\mathrm{g}} > 3 \cdot (n_{\mathrm{g}} + n_1) \cdot \lceil \log \bar{q} \rceil$, and $\bar{m} > 2 \cdot \bar{n}_{\mathrm{g}} \cdot \lceil \log \bar{q} \rceil$.
- The description of balanced admissible hash functions $\mathsf{AHF} : \{0,1\}^{\ell_t} \to \{0,1\}^L$ and $\mathsf{AHF}_f : \{0,1\}^{\ell_f} \to \{0,1\}^L$ for suitable $L \in \mathrm{poly}(\lambda)$.

- A real $\alpha > 0$ and a Gaussian parameter $\sigma > 0$ so that the interval $[-\beta, \beta] := [-\sigma\sqrt{n_{\mathrm{g}}}, \sigma\sqrt{n_{\mathrm{g}}}]$ specifies the domain for the secret vector's coordinates (with overwhelming probability).
- The real $\boxed{\mathrm{M}} := \lambda^2 \cdot \bar{\sigma}$.

We recall the specification of the gadget matrix $\bar{\mathbf{T}}$

$$\bar{\mathbf{T}} = [\mathbf{I}_{n_{\mathrm{g}}} \otimes (1, 2, 4, \ldots, 2^{\lceil \log \bar{q} \rceil}) \mid \mathbf{0}^{n_{\mathrm{g}}} \mid \cdots \mid \mathbf{0}^{n_{\mathrm{g}}}] \in \mathbb{Z}_{\bar{q}}^{n_{\mathrm{g}} \times \bar{m}} \quad .$$

The other gadgets matrices $\mathbf{T}, \mathbf{T}_0$ can be defined similarly:

$$\begin{aligned}\mathbf{T}_0 &= [\mathbf{I}_{n_0} \otimes (1, 2, 4, \ldots, 2^{\lceil \log \bar{q} \rceil}) \mid \mathbf{0}^{n_0} \mid \cdots \mid \mathbf{0}^{n_0}] \in \mathbb{Z}_q^{n_0 \times m} \\ \mathbf{T} &= [\mathbf{I}_{n_{\mathrm{g}}} \otimes (1, 2, 4, \ldots, 2^{\lceil \log \bar{q} \rceil}) \mid \mathbf{0}^{n_{\mathrm{g}}} \mid \cdots \mid \mathbf{0}^{n_{\mathrm{g}}}] \in \mathbb{Z}_q^{n_{\mathrm{g}} \times m} \quad .\end{aligned}$$

**Dynamizability.** The following Lemma 29 shows that our DMCFE scheme $\boxed{\mathcal{E}}$ can be plugged into Theorem 13 to obtain a DDFE.

**Lemma 29.** *The DMCFE scheme $\boxed{\mathcal{E}}$ in Fig. 6 is $[-M, M]^{\bar{n}_{\mathrm{g}}}$-dynamizable.*

*Proof.* The finite Abelian group $\mathbb{A}$ is $\mathbb{A} := [-M, M]^{\bar{n}_{\mathrm{g}}}$ defined for any $\bar{n}_{\mathrm{g}} \in \mathbb{N}$ and $[-M, M] := [-\lambda^2 \cdot \bar{\sigma}, \lambda^2 \cdot \bar{\sigma}]$. It suffices to specify the ppt algorithms SetupPP, SetupUser as well as the two functions $P_{\mathsf{sk}}, P_{\mathsf{ek}} \colon \{0,1\}^{\log |\mathbb{A}|} \to \{0,1\}^{\leq \log |\mathbb{A}|}$. We give the details below:

$\mathsf{SetupPP}(1^\lambda; r_0)$**:** Using the random coins $r_0$, sample the public parameters and matrices according to the security parameter $\lambda$, then output

$$\widehat{\mathsf{pp}} := \big(\mathsf{cp}, \mathbf{V}, \bar{\mathbf{V}}, (\mathbf{A}_{i,b})_{i \in [L], b \in \{0,1\}}, (\mathbf{B}_{i,b})_{i \in [L], b \in \{0,1\}}\big) \quad .$$

We note that $\mathsf{SetupPP}(1^\lambda; r_0)$ does not rely on the number of clients (or senders), which is important in our definition of dynamizability.

$\mathsf{SetupUser}(\widehat{\mathsf{pp}}; r_i)$**:** Sample $\mathbf{t}_i \overset{\$}{\leftarrow} D_{\mathbb{Z}^{\bar{n}_{\mathrm{g}}}, \bar{\sigma}}$ and $\mathbf{s}_i \overset{\$}{\leftarrow} D_{\mathbb{Z}^{n_{\mathrm{g}}}, \sigma}$ using the coins $r_i$, then define $\widehat{\mathsf{ek}}_i := \mathbf{s}_i, \widehat{\mathsf{sk}}_i := (\mathbf{s}_i, \mathbf{t}_i)$ and output $(\widehat{\mathsf{ek}}_i, \widehat{\mathsf{sk}}_i)$.

$P_{\mathsf{sk}}(s_i, \widehat{\mathsf{sk}}_i)$**:** Given inputs $\widehat{\mathsf{sk}}_i = (\mathbf{s}_i, \mathbf{t}_i)$ and $s_i = \mathbf{v}_i$ such that $(\mathbf{v}_i)_{i \in \bar{n}_{\mathrm{g}}} \leftarrow \mathcal{S}(n, \mathbb{A}; r_{\mathcal{S}})$, *i.e.* $(\mathbf{v}_i)_{i \in \bar{n}_{\mathrm{g}}}$ is uniformly random shares of $\mathbf{0}$ in $\mathbb{A}$ following a ppt algorithm with random coins $r_{\mathcal{S}}$, output $(\mathbf{s}_i, \mathbf{t}_i, \mathbf{u}_i := \mathbf{t}_i + \mathbf{v}_i)$.

$P_{\mathsf{ek}}(s_i, \widehat{\mathsf{ek}}_i)$**:** Given inputs $\widehat{\mathsf{ek}}_i$ and $s_i = \mathbf{v}_i$ belonging to $(\mathbf{v}_i)_{i \in \bar{n}_{\mathrm{g}}} \leftarrow \mathcal{S}(n, \mathbb{A}; r_{\mathcal{S}})$, output $\widehat{\mathsf{ek}}_i = \mathbf{s}_i$.

By construction, for any $\lambda, n \in \mathbb{N}$, the distribution $\left\{\widehat{\mathsf{pp}}, \big(P_{\mathsf{sk}}(s_i, \widehat{\mathsf{sk}}_i), P_{\mathsf{ek}}(s_i, \widehat{\mathsf{ek}}_i)\big)_{i \in [\ell]}\right\}$ is identical to $\{\mathsf{pp}, (\mathsf{sk}_i, \mathsf{ek}_i)_{i \in [\ell]}\}$ from $\boxed{\mathcal{E}}$ in Fig. 6. $\square$

**Security.** We now prove the security of our DMCFE scheme $\boxed{\mathcal{E}}$ from Figure 6. The main theorem is stated below.

**Theorem 30.** *If the DMCFE scheme $\mathcal{E}$ in [45] is secure against* adaptive *key-generation and encryption queries under* static *corruption, then so is our modified scheme $\boxed{\mathcal{E}}$ in Fig. 6.*

*Proof.* Let $\mathcal{A}$ be an adversary in the experiment $\mathbf{Exp}^{\mathsf{stat\text{-}cpa}}_{\boxed{\mathcal{E}}, \mathcal{F}^{\mathsf{ip}}, \mathcal{A}}(1^\lambda)$ attacking the security of $\boxed{\mathcal{E}}$. We build an adversary $\mathcal{B}$ for the experiment $\mathbf{Exp}^{\mathsf{stat\text{-}cpa}}_{\mathcal{E}, \mathcal{F}^{\mathsf{ip}}, \mathcal{B}}(1^\lambda)$ that attacks the security of the original scheme $\mathcal{E}$. For convenience, we introduce the shorthands

$$\begin{aligned}\left(\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}, \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{A}}\right) &:= \left(\mathbf{Exp}^{\mathsf{stat\text{-}cpa}}_{\boxed{\mathcal{E}}, \mathcal{F}^{\mathsf{ip}}, \mathcal{A}}(1^\lambda), \mathbf{Adv}^{\mathsf{stat\text{-}cpa}}_{\boxed{\mathcal{E}}, \mathcal{F}^{\mathsf{ip}}, \mathcal{A}}(1^\lambda)\right) \\ \left(\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}, \mathbf{Adv}^{\mathsf{ip}}_{\mathcal{B}}\right) &:= \left(\mathbf{Exp}^{\mathsf{stat\text{-}cpa}}_{\mathcal{E}, \mathcal{F}^{\mathsf{ip}}, \mathcal{B}}(1^\lambda), \mathbf{Adv}^{\mathsf{stat\text{-}cpa}}_{\mathcal{E}, \mathcal{F}^{\mathsf{ip}}, \mathcal{B}}(1^\lambda)\right) \quad .\end{aligned}$$

The adversary $\mathcal{B}$ works as follows:

$\mathsf{Setup}(\mathsf{cp}, 1^n)$**:** On input common global public parameters $\mathsf{cp}$, the dimension $n$ in unary, sample random matrices:

$$\mathbf{V} \in \mathbb{Z}_q^{n_0 \times n_{\mathrm{g}}}, \bar{\mathbf{V}} \in \mathbb{Z}_q^{n \times \bar{n}_{\mathrm{g}}}, \left(\mathbf{A}_{i,b} \in \mathbb{Z}_q^{n_{\mathrm{g}} \times m}\right)_{i\in[L], b\in\{0,1\}}, \left(\mathbf{B}_{i,b} \in \mathbb{Z}_{\bar{q}}^{\bar{n}_{\mathrm{g}} \times \bar{m}}\right)_{i\in[L], b\in\{0,1\}}$$

as well as gaussian samples $\mathbf{t}_i \xleftarrow{\$} D_{\mathbb{Z}^{\bar{n}_{\mathrm{g}}}, \bar{\sigma}}, \mathbf{s}_i \xleftarrow{\$} D_{\mathbb{Z}^{n_{\mathrm{g}}}, \sigma}$ for $i \in [n]$. The sum of $\mathbf{t}_i$ is denoted by $\mathbf{t} = \sum_{i=1}^n \mathbf{t}_i$. $\boxed{\text{For each } i \in [\ell], \text{ sample } \mathbf{v}_i \xleftarrow{\$} [-M, M]^{\bar{n}_{\mathrm{g}}} \text{ such that } \sum_{i\in[\ell]} \mathbf{v}_i = 0.}$ Output

$$\mathsf{pp} := \left(\mathsf{cp}, \mathbf{V}, \bar{\mathbf{V}}, (\mathbf{A}_{i,b})_{i\in[L],b\in\{0,1\}}, (\mathbf{B}_{i,b})_{i\in[L],b\in\{0,1\}},\ \mathbf{t}\right)\ ,$$
$$\boxed{:= \left(\mathsf{cp}, \mathbf{V}, \bar{\mathbf{V}}, (\mathbf{A}_{i,b})_{i\in[L],b\in\{0,1\}}, (\mathbf{B}_{i,b})_{i\in[L],b\in\{0,1\}}\right)\ ,}$$
$$\left(\mathsf{ek}_i := \mathbf{s}_i, \mathsf{sk}_i := (\mathbf{s}_i, \mathbf{t}_i \boxed{, \mathbf{u}_i := \mathbf{t}_i + \mathbf{v}_i})\right)_{i\in[\ell]}\ .$$

$\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, y_i)$**:** On input a secret key $\mathsf{sk}_i = (\mathbf{s}_i \in \mathbb{Z}^{n_{\mathrm{g}}}, \mathbf{t}_i \in \mathbb{Z}^{\bar{n}_{\mathrm{g}}} \boxed{, \mathbf{u}_i})$, a function tag $\mathsf{tag\text{-}f}$ and a scalar $y_i \in \mathsf{Param}_\lambda$, where $\mathsf{Param}_\lambda = [-Y, Y]$, compute

1. The hash $\tau_{\mathsf{tag\text{-}f}} = \tau_{\mathsf{tag\text{-}f}}[1] \dots \tau_{\mathsf{tag\text{-}f}}[L] := \mathsf{AHF_f}(\mathsf{tag\text{-}f}) \in \{0,1\}^L$ as well as the GSW evaluation

$$\mathbf{B}(\tau_{\mathsf{tag\text{-}f}}) = \mathbf{B}_{L,\tau_{\mathsf{tag\text{-}f}}[L]} \cdot \bar{\mathbf{T}}^{-1}\left(\mathbf{B}_{L-1,\tau_{\mathsf{tag\text{-}f}}[L-1]} \cdot \bar{\mathbf{T}}^{-1}(\cdots \mathbf{B}_{2,\tau_{\mathsf{tag\text{-}f}}[2]} \cdot \bar{\mathbf{T}}^{-1}(\mathbf{B}_{1,\tau_{\mathsf{tag\text{-}f}}[1]}))\right)$$
$$\cdot\, \bar{\mathbf{T}}^{-1}(\bar{\mathbf{W}}^\top) \in \mathbb{Z}_{\bar{q}}^{\bar{n}_{\mathrm{g}} \times \bar{m}} \quad (11)$$

   and $\bar{\mathbf{W}} = \bar{\mathbf{T}}^\top \cdot \bar{\mathbf{V}} \in \mathbb{Z}_{\bar{q}}^{\bar{m} \times \bar{n}_{\mathrm{g}}}$.

2. Sample $\mathbf{e}_{\mathsf{tag\text{-}f},i} \xleftarrow{\$} D_{\bar{m},\alpha\bar{q}}$ and output $\mathsf{dk}_i = \left(\mathbf{d}_i := \bar{\mathbf{T}}^\top \cdot (y_i \cdot \mathbf{s}_i) + \mathbf{B}(\tau_{\mathsf{tag\text{-}f}})^\top \cdot \mathbf{t}_i + \mathbf{e}_{\mathsf{tag\text{-}f},i} \in \mathbb{Z}_{\bar{q}}^{\bar{m}} \boxed{, \mathbf{u}_i}\right)$.

$\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$**:** On input an encryption key $\mathsf{ek}_i = \mathbf{s}_i \in \mathbb{Z}^{n_{\mathrm{g}}}$, a tag $\mathsf{tag}$ and a vector $\mathbf{x}_i \in [-X, X]^{n_0}$, compute

1. The hash $\tau_{\mathsf{tag}} = \tau_{\mathsf{tag}}[1] \dots \tau_{\mathsf{tag}}[L] := \mathsf{AHF}(\mathsf{tag}) \in \{0,1\}^L$ as well as the GSW evaluation

$$\mathbf{A}(\tau_{\mathsf{tag}}) = \mathbf{A}_{L,\tau_{\mathsf{tag}}[L]} \cdot \mathbf{T}^{-1}\left(\mathbf{A}_{L-1,\tau_{\mathsf{tag}}[L-1]} \cdot \mathbf{T}^{-1}(\cdots \mathbf{A}_{2,\tau_{\mathsf{tag}}[2]} \cdot \mathbf{T}^{-1}(\mathbf{A}_{1,\tau_{\mathsf{tag}}[1]}))\right)$$
$$\cdot\, \mathbf{T}^{-1}(\mathbf{W}^\top) \in \mathbb{Z}_q^{n_{\mathrm{g}} \times m} \quad (12)$$

   and $\mathbf{W} = \mathbf{T}_0^\top \cdot \mathbf{V} \in \mathbb{Z}_q^{m \times n_{\mathrm{g}}}$.

2. Sample $\mathbf{e}_{\mathsf{tag},i} \xleftarrow{\$} D_{\mathbb{Z}^m,\alpha q}$ and output $\mathsf{ct}_i = \mathbf{T}_0^\top \cdot \mathbf{x}_i + \mathbf{A}(\tau_{\mathsf{tag}})^\top \cdot \mathbf{s}_i + \mathbf{e}_{\mathsf{tag},i}$.

$\mathsf{Dec}((\mathsf{dk}_i)_{i\in[\ell]}, (\mathsf{ct}_i)_{i\in[\ell]})$**:** On input a list of decryption keys $(\mathsf{dk}_i = (\mathbf{d}_i, \boxed{\mathbf{u}_i}))_{i\in[\ell]}$ and a list of ciphertexts $(\mathsf{ct}_i)_{i\in[\ell]}$, perform the key combination by $\boxed{\text{first computing } \mathbf{t} = \sum_{i\in[\ell]} \mathbf{u}_i \in \mathbb{Z}^{\bar{n}_{\mathrm{g}}},}$ $\tau_{\mathsf{tag\text{-}f}} = \tau_{\mathsf{tag\text{-}f}}[1] \dots \tau_{\mathsf{tag\text{-}f}}[L] := \mathsf{AHF_f}(\mathsf{tag\text{-}f}) \in \{0,1\}^L$, and $\widetilde{\mathbf{d}}_f := \sum_{i\in[\ell]} \mathsf{dk}_i - \mathbf{B}(\tau_{\mathsf{tag\text{-}f}})^\top \cdot \mathbf{t} \mod \bar{q}$, where $\mathbf{B}(\tau_{\mathsf{tag\text{-}f}})$ is get *as per* (11). Interpret $\widetilde{\mathbf{d}}_f = \bar{\mathbf{T}}^\top \mathbf{d}_f + \widetilde{\mathbf{e}}_f$, use the public trapdoor of $\Lambda^\perp(\bar{\mathbf{T}})$ to compute $\mathbf{d}_f$. Next, compute $\tau_{\mathsf{tag}} = \tau_{\mathsf{tag}}[1] \dots \tau_{\mathsf{tag}}[L] := \mathsf{AHF}(\mathsf{tag}) \in \{0,1\}^L$, then $\mathbf{z}_{\mathsf{tag}} = \sum_{i\in[\ell]} y_i \cdot \mathsf{ct}_i - \mathbf{A}(\tau_{\mathsf{tag}})^\top \cdot \mathbf{d}_f \mod q$, where $\mathbf{A}(\tau_{\mathsf{tag}})$ is computed *as per* (12). Interpret $\mathbf{z}_{\mathsf{tag}} = \mathbf{T}_0^\top \mathbf{z} + \mathbf{e}$, use the public trapdoor of $\Lambda^\perp(\mathbf{T}_0)$ to compute $\mathbf{z} \in [-\ell XY, \ell XY]$.

Fig. 6: The $\mathsf{DMCFE}$ scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ given in [45] and our variant $\boxed{\mathcal{E}}$ with differing details in $\boxed{\text{boxes}}$.

- *Initialization and Static Corruption Queries:* Upon $\mathcal{A}$ calling $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}.\mathsf{Initialize}(1^\lambda)$, $\mathcal{B}$ calls $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathsf{Initialize}(1^\lambda)$ to obtain

$$\mathsf{pp} = \left(\mathsf{cp}, \mathbf{V}, \bar{\mathbf{V}}, (\mathbf{A}_{i,b})_{i\in[L],b\in\{0,1\}}, (\mathbf{B}_{i,b})_{i\in[L],b\in\{0,1\}}, \mathbf{t}\right)\ .$$

  and sends $\boxed{\mathsf{pp}} = \left(\mathsf{cp}, \mathbf{V}, \bar{\mathbf{V}}, (\mathbf{A}_{i,b})_{i\in[L],b\in\{0,1\}}, (\mathbf{B}_{i,b})_{i\in[L],b\in\{0,1\}}\right)$ to $\mathcal{A}$. The adversary $\mathcal{B}$ aborts if $\mathbf{t} \notin [-M, M]^{\bar{n}_{\mathrm{g}}}$.
  In the *static* corruption setting, $\mathcal{A}$ sends up front a set $\mathcal{C} \subset [\ell]$ of corrupted $i$ to $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}.\mathcal{O}\mathsf{Corrupt}$. For each $i \in \mathcal{C}$, $\mathcal{B}$ samples $\mathbf{v}_i \xleftarrow{\$} [-M, M]^{\bar{n}_{\mathrm{g}}}$, queries $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{Corrupt}(i)$ to receive $(\mathbf{s}_i \in \mathbb{Z}^{n_{\mathrm{g}}}, \mathbf{t}_i \in \mathbb{Z}^{\bar{n}_{\mathrm{g}}})$, defines $\mathbf{u}_i := \mathbf{t}_i + \mathbf{v}_i$ and sends $\{(\mathsf{ek}_i := \mathbf{s}_i, \mathsf{sk}_i = (\mathbf{s}_i, \mathbf{t}_i, \mathbf{u}_i))\}_{i\in\mathcal{C}}$ to $\mathcal{A}$. Furthermore, for each $i \in \mathcal{H} := [\ell] \setminus \mathcal{C}$, $\mathcal{B}$ samples vectors $\mathbf{u}_i \xleftarrow{\$} [-M, M]^{\bar{n}_{\mathrm{g}}}$ conditioned on $\sum_{i\in\mathcal{H}} \mathbf{u}_i = \mathbf{t} - \sum_{i\in\mathcal{C}}(\mathbf{t}_i + \mathbf{v}_i)$.

- *Encryption Queries:* W.l.o.g, we assume that $\mathcal{A}$ will query $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}.\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ only for $i \in \mathcal{H}$. In that case $\mathcal{B}$ queries $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ and forwards the result to $\mathcal{A}$.
- *Key-Generation Queries:* W.l.o.g, we assume that $\mathcal{A}$ will query $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}.\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}, \mathbf{y}_i^{(0)}, \mathbf{y}_i^{(1)})$ only for $i \in \mathcal{H}$. Then, $\mathcal{B}$ calls $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathcal{O}\mathsf{DKeyGen}(i, \mathsf{tag\text{-}f}, \mathbf{y}_i^{(0)}, \mathbf{y}_i^{(1)})$ to obtain $\mathbf{d}_i$ and sends $\boxed{\mathsf{dk}_i} = (\mathbf{d}_i, \mathbf{u}_i)$ to $\mathcal{A}$.
- *Finalize:* Upon $\mathcal{A}$ calling $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}.\mathsf{Finalize}(b')$, $\mathcal{B}$ calls $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{B}}.\mathsf{Finalize}(b')$.

For $\mathbf{t} = \sum_{i\in[\ell]} \mathbf{t}_i$ with $\mathbf{t}_i \xleftarrow{\$} D_{\mathbb{Z}^{\bar{n}_\mathsf{g}},\bar{\sigma}}$ *i.i.d*, it holds that $\mathbf{t}$ follows the Gaussian distribution $D_{\mathbb{Z}^{\bar{n}_\mathsf{g}},\ell\cdot\bar{\sigma}}$ where the standard deviation is multiplied by a factor $\ell$. By using the fact that the center of $D_{\mathbb{Z}^{\bar{n}_\mathsf{g}},\ell\cdot\bar{\sigma}}$ is $\mathbf{0}$ together with the union bound, the Chernoff-Cramér bound (Theorem 28) applied for Gaussian random variables yields:

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

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

$$D_0 := \left\{ \begin{array}{l} \{(\mathbf{t}_i, \mathbf{v}_i)\}_{i\in\mathcal{C}}; \\ \{\mathbf{u}_i\}_{i\in\mathcal{H}}; \mathbf{t} \end{array} \;\middle|\; \begin{array}{l} \forall i \in [\ell] : \mathbf{t}_i \xleftarrow{\$} D_{\mathbb{Z}^{\bar{n}_\mathsf{g}},\bar{\sigma}};\ \mathbf{t} := \sum_{i\in[\ell]} \mathbf{t}_i \\ \forall i \in \mathcal{C} : \mathbf{v}_i \xleftarrow{\$} [-M, M]^{\bar{n}_\mathsf{g}} \\ \boxed{\begin{array}{l} \forall i \in \mathcal{H} : \mathbf{u}_i \xleftarrow{\$} [-M, M]^{\bar{n}_\mathsf{g}} \text{ s.t.} \\ \qquad\quad \sum_{i\in\mathcal{H}} \mathbf{u}_i = \mathbf{t} - \sum_{i\in\mathcal{C}}(\mathbf{t}_i + \mathbf{v}_i) \end{array}} \end{array} \right\}$$

$$D_1 := \left\{ \begin{array}{l} \{(\mathbf{t}_i, \mathbf{v}_i)\}_{i\in\mathcal{C}}; \\ \{\mathbf{u}_i\}_{i\in\mathcal{H}}; \mathbf{t} \end{array} \;\middle|\; \begin{array}{l} \forall i \in [\ell] : \mathbf{t}_i \xleftarrow{\$} D_{\mathbb{Z}^{\bar{n}_\mathsf{g}},\bar{\sigma}};\ \mathbf{t} := \sum_{i\in[\ell]} \mathbf{t}_i \\ \forall i \in [\ell] : \mathbf{v}_i \xleftarrow{\$} [-M, M]^{\bar{n}_\mathsf{g}} \\ \boxed{\forall i \in \mathcal{H} : \mathbf{u}_i := \mathbf{t}_i + \mathbf{v}_i} \end{array} \right\},$$

where $D_0$ corresponds to the simulation of $\mathcal{B}$ and $D_1$ corresponds to the responses in the experiment $\mathbf{Exp}^{\mathsf{ip}}_{\mathcal{A}}$.
We recall that $\mathbf{U}(S)$ denotes the uniform distribution on a finite set $S$, and that $\mathcal{S}(\ell, [-M, M]^{\bar{n}_\mathsf{g}})$ denotes the distribution that outputs $\mathbf{v}_i \xleftarrow{\$} [-M, M]^{\bar{n}_\mathsf{g}}$ for $i \in [\ell]$ conditioned on $\sum_{i\in[\ell]} \mathbf{v}_i = 0$. W.l.o.g, we extend $U([-M, M]^{\bar{n}_\mathsf{g}})$ over $[-M, M]^{\bar{n}_\mathsf{g}}$ to over $\mathbb{Z}^{\bar{n}_\mathsf{g}}$ such that for any $\mathbf{x} \in \mathbb{Z}^{\bar{n}_\mathsf{g}} \setminus [-M, M]^{\bar{n}_\mathsf{g}}$ it holds $\Pr_{X\sim U([-M,M]^{\bar{n}_\mathsf{g}})}[X = \mathbf{x}] = 0$. For the ease of notation, we use boldface letters $(\mathbf{T}, \mathbf{U}, \mathbf{V})$ to denote collections of vectors sampled following a given distribution.
We will make use of the following lemma which is proven below.

**Lemma 31.** *For each $\mathbf{U} \in ([-M, M]^{\bar{n}_\mathsf{g}})^{|\mathcal{H}|}$, we have*

$$\Pr[D_1 \to \mathbf{U}] \leq \frac{(2M+1)^{\bar{n}_\mathsf{g}}}{(2M+1)^{\bar{n}_\mathsf{g}|\mathcal{H}|}} \left(1 + \frac{2|\mathcal{H}|\pi\bar{n}_\mathsf{g}B^2}{2\bar{\sigma}^2 - |\mathcal{H}|\pi\bar{n}_\mathsf{g}B^2}\right) + \left(\frac{1}{(2M+1)^{\bar{n}_\mathsf{g}}}\right)^{|\mathcal{H}|-1},$$

*where $B := \lambda \cdot \bar{\sigma}$.*

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

$$\begin{aligned}
&\varDelta(D_0, D_1) && (13)\\
&\overset{(1)}{\leq} \max_{\mathcal{S}\subseteq([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}} |\Pr[D_1 \in \mathcal{S}] - \Pr[D_0 \in \mathcal{S}]| + \mathrm{negl}_1(\lambda)\\
&\overset{(2)}{\leq} \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \forall\mathbf{U}\in\mathcal{S}:\Pr[D_1\to\mathbf{U}]>\Pr[D_0\to\mathbf{U}]}} \left(\Pr[D_1 \to \mathbf{U}] - \left(\frac{1}{(2M+1)^{\bar{n}_\mathrm{g}}}\right)^{|\mathcal{H}|-1}\right) + \mathrm{negl}_1(\lambda) && (14)
\end{aligned}$$

We note that (1) follows from the definition of the statistical distance and Theorem 28, and (2) applies the uniform choice of $\mathbf{U}$ in $D_0$. Next, we use Lemma 31 and obtain from (14) that

$$\begin{aligned}
&\varDelta(D_0, D_1)\\
&\leq \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \forall\mathbf{U}\in\mathcal{S}:\Pr[D_1\to\mathbf{U}]>\Pr[D_0\to\mathbf{U}]}} \left(\frac{(2M+1)^{\bar{n}_\mathrm{g}}}{(2M+1)^{\bar{n}_\mathrm{g}|\mathcal{H}|}} \cdot \left(1 + \frac{2|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}{2\bar{\sigma}^2 - |\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}\right)\right.\\
&\qquad\qquad \left. + \left(\frac{1}{(2M+1)^{\bar{n}_\mathrm{g}}}\right)^{|\mathcal{H}|-1} - \left(\frac{1}{(2M+1)^{\bar{n}_\mathrm{g}}}\right)^{|\mathcal{H}|-1}\right) + \mathrm{negl}_1(\lambda)\\
&\leq \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \forall\mathbf{U}\in\mathcal{S}:\Pr[D_1\to\mathbf{U}]>\Pr[D_0\to\mathbf{U}]}} \frac{(2M+1)^{\bar{n}_\mathrm{g}}}{(2M+1)^{\bar{n}_\mathrm{g}|\mathcal{H}|}} \cdot \left(1 + \frac{2|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}{2\bar{\sigma}^2 - |\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}\right) + \mathrm{negl}_1(\lambda)\\
&\overset{(3)}{\leq} \sum_{\substack{\mathbf{U}\in\mathcal{S}\subseteq([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \forall\mathbf{U}\in\mathcal{S}:\Pr[D_1\to\mathbf{U}]>\Pr[D_0\to\mathbf{U}]}} \frac{1}{(2M+1)^{\bar{n}_\mathrm{g}|\mathcal{H}|}} \cdot \exp\left(2M\bar{n}_\mathrm{g} - \frac{2|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}{|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2 - 2\bar{\sigma}^2}\right) + \mathrm{negl}_1(\lambda)\\
&\leq \exp\left(2M\bar{n}_\mathrm{g} - \frac{2|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}{|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2 - 2\bar{\sigma}^2}\right) + \mathrm{negl}_1(\lambda) \ ,
\end{aligned}$$

where (3) uses the fact that $1 + x \leq e^x$ from (10). To bound $\mathrm{negl}_1(\lambda)$, we can perform a similar calculation as it is done for $\mathrm{negl}_2(\lambda)$ in the proof of Lemma 31. By parameter choices $|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2 = \omega(1)\bar{\sigma}^2$ and $M = o(|\mathcal{H}|\pi B^2)$, then $\exp\left(2M\bar{n}_\mathrm{g} - \frac{2|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2}{|\mathcal{H}|\pi\bar{n}_\mathrm{g}B^2-2\bar{\sigma}^2}\right)$ is negligible in $\lambda$ and the proof is completed. □

We now prove the lemma.

*Proof (of Lemma 31).* For $\mathbf{U} \in ([-M, M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}$, noting that in $D_1$ each $\mathbf{t}_i \xleftarrow{\$} D_{\mathbb{Z}^{\bar{n}_\mathrm{g}},\bar{\sigma}}$ is *i.i.d* by construction, we compute

$$\begin{aligned}
&\Pr[D_1 \to \mathbf{U}]\\
&\leq \sum_{\substack{\mathbf{T}\in([-B,B]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \mathbf{V}\in([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \mathbf{T}+\mathbf{V}=\mathbf{U}}} \Pr[\forall i \in \mathcal{H} : D_{\mathbb{Z}^{\bar{n}_\mathrm{g}},\bar{\sigma}} \to \mathbf{T}[i]] \cdot \Pr[\mathcal{S}(n, [-M, M]^{\bar{n}_\mathrm{g}}) \to \mathbf{V}] + \mathrm{negl}_2(\lambda)\\
&= \sum_{\substack{\mathbf{T}\in([-B,B]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \mathbf{V}\in([-M,M]^{\bar{n}_\mathrm{g}})^{|\mathcal{H}|}\\ \mathbf{T}+\mathbf{V}=\mathbf{U}}} \left(\prod_{i=1}^{|\mathcal{H}|} \frac{\rho_{\bar{\sigma}}(\mathbf{T}[i])}{\rho_{\bar{\sigma}}(\mathbb{Z}^{\bar{n}_\mathrm{g}})}\right) \cdot \left(\frac{1}{(2M+1)^{\bar{n}_\mathrm{g}}}\right)^{|H|-1} + \mathrm{negl}_2(\lambda) && (15)
\end{aligned}$$

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

$$\begin{aligned}
\frac{\rho_{\bar{\sigma}}(\mathbf{T}[i])}{\rho_{\bar{\sigma}}(\mathbb{Z}^{\bar{n}_{\mathrm{g}}})} \leq \frac{\rho_{\bar{\sigma}}(\mathbf{T}[i])}{\rho_{\bar{\sigma}}([-B, B]^{\bar{n}_{\mathrm{g}}})} &= \frac{\exp(-\pi \mathbf{T}[i]^\top \mathbf{\Sigma}^{-1} \mathbf{T}[i])}{\sum_{\mathbf{T} \in [-B,B]^{\bar{n}_{\mathrm{g}}}} \exp(-\pi \mathbf{T}^\top \mathbf{\Sigma}^{-1} \mathbf{T})} \\
&= \frac{\exp(-\pi \|\mathbf{T}[i]\|_2^2 / \bar{\sigma}^2)}{\sum_{\widehat{\mathbf{T}} \in [-B,B]^{\bar{n}_{\mathrm{g}}}} \exp(-\pi \|\widehat{\mathbf{T}}\|_2^2 / \bar{\sigma}^2)} \\
&\overset{(1)}{\leq} \frac{\exp(\pi \bar{n}_{\mathrm{g}} B^2 / \bar{\sigma}^2)}{(2B+1)^{\bar{n}_{\mathrm{g}}}} \\
&\overset{(2)}{\leq} \frac{\left(1 + \frac{2|\mathcal{H}|\pi \bar{n}_{\mathrm{g}} B^2}{2\bar{\sigma}^2 - |\mathcal{H}|\pi \bar{n}_{\mathrm{g}} B^2}\right)^{1/|\mathcal{H}|}}{(2B+1)^{\bar{n}_{\mathrm{g}}}} \quad , \qquad (16)
\end{aligned}$$

where (1) follows from the fact that the squared Euclidean norm $\|\mathbf{T}[i]\|_2^2$ is bounded by $\bar{n}_{\mathrm{g}} B^2$, and (2) uses the inequality $e^x \leq \left(1 + \frac{x}{n}\right)^{n+x/2}$ from (10). We observe that any choice of $\mathbf{T}$ given $\mathbf{U}$ fixes $\mathbf{V}$. Then plugging (16) into (15) implies

$$\begin{aligned}
\Pr[D_1 \to \mathbf{U}] &\leq \left(\frac{2B+1}{2M+1}\right)^{\bar{n}_{\mathrm{g}} \cdot |\mathcal{H}|} \cdot (2M+1)^{\bar{n}_{\mathrm{g}}} \left(\frac{1}{(2B+1)^{\bar{n}_{\mathrm{g}}}}\right)^{|\mathcal{H}|} \cdot \left(1 + \frac{2|\mathcal{H}|\pi \bar{n}_{\mathrm{g}} B^2}{2\bar{\sigma}^2 - |\mathcal{H}|\pi \bar{n}_{\mathrm{g}} B^2}\right) + \mathrm{negl}_2(\lambda) \\
&\leq \frac{(2M+1)^{\bar{n}_{\mathrm{g}}}}{(2M+1)^{\bar{n}_{\mathrm{g}}|\mathcal{H}|}} \left(1 + \frac{2|\mathcal{H}|\pi \bar{n}_{\mathrm{g}} B^2}{2\bar{\sigma}^2 - |\mathcal{H}|\pi \bar{n}_{\mathrm{g}} B^2}\right) + \mathrm{negl}_2(\lambda) \quad . \qquad (17)
\end{aligned}$$

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

$$\begin{aligned}
\mathrm{negl}_2(\lambda) &\leq \sum_{\mathbf{V} \in ([-M,M])^{|\mathcal{H}|}} \Pr[\mathcal{S}(\ell, [-M, M]^{\bar{n}_{\mathrm{g}}}) \to \mathbf{V}] \cdot \Pr[\mathbf{T} \notin ([-B, B]^{\bar{n}_{\mathrm{g}}})^{|\mathcal{H}|}] \\
&\overset{(3)}{\leq} (2M+1)^{\bar{n}_{\mathrm{g}}|\mathcal{H}|} \left(\frac{1}{(2M+1)^{\bar{n}_{\mathrm{g}}}}\right)^{|\mathcal{H}|-1} \cdot \Pr[\mathbf{T} \notin ([-B, B]^{\bar{n}_{\mathrm{g}}})^{|\mathcal{H}|}] \\
&\overset{(4)}{\leq} (2M+1)^{\bar{n}_{\mathrm{g}}|\mathcal{H}|} \left(\frac{1}{(2M+1)^{\bar{n}_{\mathrm{g}}}}\right)^{|\mathcal{H}|-1} \cdot 2\bar{n}_{\mathrm{g}}|\mathcal{H}| \cdot \exp\left(-\frac{B^2}{2\bar{\sigma}^2}\right) \\
&\overset{(5)}{\leq} \exp\left(2M\bar{n}_{\mathrm{g}}|\mathcal{H}|\right) \cdot \left(\frac{1}{(2M+1)^{\bar{n}_{\mathrm{g}}}}\right)^{|\mathcal{H}|-1} \cdot 2\bar{n}_{\mathrm{g}}|\mathcal{H}| \cdot \exp\left(-\frac{B^2}{2\bar{\sigma}^2}\right) \\
&= \left(\frac{1}{(2M+1)^{\bar{n}_{\mathrm{g}}}}\right)^{|\mathcal{H}|-1} \cdot 2\bar{n}_{\mathrm{g}}|\mathcal{H}| \cdot \exp\left(2M\bar{n}_{\mathrm{g}}|\mathcal{H}| - \frac{B^2}{2\bar{\sigma}^2}\right) \\
&\overset{(6)}{\leq} \left(\frac{1}{(2M+1)^{\bar{n}_{\mathrm{g}}}}\right)^{|\mathcal{H}|-1} \quad ,
\end{aligned}$$

where (3) uses the union bound over all possible values of $\mathbf{V}$ and the fact that it is uniformly distributed, (4) employs the Gaussian distribution of $\mathbf{T}$ and Theorem 28, (5) uses the inequality $1 + x \leq e^x$ from (10) and (6) follows from the parameter choice $M = o(B^2)$. This concludes the proof. □

## C Supporting Materials – A FH-DMCFE for Inner Products

### C.1 Construction

This section presents an FH-DMCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for the function class $\mathcal{F}^{\mathsf{ip}}$. We note that each client encrypts a vector padded to the same length of $N \in \mathbb{N}$. We work in

$\mathsf{Setup}(1^\lambda, 1^n)$**:** Run $\mathbb{G} \leftarrow \mathsf{GGen}(1^\lambda)$ and $\mathsf{DPVSGen}(\mathbb{G}, 1^{4N+5})$ for $i \in [n]$ to obtain $n$ pairs of matrices $(B_i, B_i^*)$ of dimensions $4N+5$ that specify dual orthogonal bases $(\mathbf{B}_i, \mathbf{B}_i^*)$.[a] Sample $(\tilde{s}_i)_i, (\tilde{t}_i)_i \xleftarrow{\$} \mathbb{Z}_q^n$ conditioned on $\sum_{i=1}^n \tilde{s}_i = \sum_{i=1}^n \tilde{t}_i = 0$. Then, output the *public parameters* $\mathsf{pp} := \mathbb{G}$, *secret keys* $\mathsf{sk}_i$ and the *encryption keys* $\mathsf{ek}_i$ as follows:

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

$\mathsf{DKeyGen}(\mathsf{sk}_i, \mathsf{tag\text{-}f}, \mathbf{y}_i)$**:** Parse $\mathsf{sk}_i = (\tilde{s}_i, (\mathbf{b}^*_{i,1}, \ldots, \mathbf{b}^*_{i,N}, B^*_{i,N+1}, B^*_{i,N+2}, \mathbf{b}^*_{i,N+3}))$. 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

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

$\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$**:** Parse $\mathsf{ek}_i = (\tilde{t}_i, (\mathbf{b}_{i,1}, \ldots, \mathbf{b}_{i,N}, B_{i,N+1}, B_{i,N+2}, \mathbf{b}_{i,N+4}))$. 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

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

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

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

Fig. 7: $\mathsf{FH\text{-}DMCFE}$ scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ for inner products. We work in the prime-order bilinear group setting $\mathbb{G} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$. We use 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 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 $\mathcal{E}$ are given in Fig. 7.

**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.** The formal definition of security notions for $\mathsf{FH\text{-}DMCFE}$ is recalled in Definition 21 in Appendix A.3. Theorem 32 states that the scheme is *weakly function-hiding*, *one-challenge* secure against *complete queries* under *static corruption*. Below, we argue that most restrictions on the security model can be removed by applying a sequence of generic lemmas.

**Theorem 32.** *The DMCFE scheme $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ in Fig. 7 for the function class $\mathcal{F}^{\mathsf{ip}}$ is* one-challenge, weakly function-hiding *secure against* complete *queries under* static *corruption in the ROM, if the* SXDH *assumption holds for $(\mathbb{G}_1, \mathbb{G}_2)$.*
*More specifically, we let $q_e$ and $q_k$ denote the maximum number of distinct tags queried to $\mathcal{O}\mathsf{Enc}$ and $\mathcal{O}\mathsf{KeyGen}$, respectively. Furthermore, for $i \in [n]$ and $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$, we define $J_{i,\mathsf{tag}}$ and $\tilde{J}_{i,\mathsf{tag\text{-}f}}$ to be the numbers of queries of the form $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, \star, \star)$ and $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag\text{-}f}, \star, \star)$, and we set $J := \max_{i\in[n],\mathsf{tag}\in\mathsf{Tag}} J_{i,\mathsf{tag}}$ and $\tilde{J} := \max_{i\in[n],\mathsf{tag\text{-}f}\in\mathsf{Tag}} \tilde{J}_{i,\mathsf{tag\text{-}f}}$, respectively. Then, for any* ppt *adversary $\mathcal{A}$ against $\mathcal{E}$, we have the following bound:*

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

The proof can be found in Appendix C.3, whose ideas are presented in Section 3.1. In Section 2.3, we argue that DMCFE can be viewed as a special case of DDFE. Therefore, it is possible to apply generic lemmas stated in the DDFE setting to DMCFE schemes, too. Specifically, we first apply Lemma 14 to allow *incomplete* queries, then Lemma 5 to allow *multiple* challenges, and finally Lemma 6 to guarantee *(full-fledged) function-hiding*. In this way, we obtain a FH-DMCFE for inner products whose only constraint on the security model is static corruption.

**Corollary 33.** *There exists an FH-DMCFE scheme for the function class $\mathcal{F}^{\mathsf{ip}}$ that is* function-hiding *secure against* static *corruption under the* SXDH *assumption in the ROM.*

## C.2 Swapping Lemma

In this section we state a technical lemma that will be the basis of the security analysis of our function-hiding IP-DMCFE. An overview on how this lemma is used can be revisited in Section 3.1. The proof is divided into three parts. We start with an informal description of the main ideas in Section C.4. Then we formally prove an important special case in Section C.5, followed by a full proof of the general lemma in Section C.6.

**Lemma 34 (Swapping).** *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda), J_i = J_i(\lambda), \widetilde{J}_i = \widetilde{J}_i(\lambda), N = N(\lambda) \in \mathbb{N}$ where $i \in [H]$ and $H, K, L, J_i, \widetilde{J}_i, N : \mathbb{N} \to \mathbb{N}$ are polynomials. Let $(\mathbf{B}_i, \mathbf{B}_i^*)$, for each $i \in [H]$, be a pair of random dual bases of dimension $4N+4$ 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. Let $R, R_1, \ldots, R_K \in \mathbb{Z}_q$ be some public scalars. For $i \in [H]$, $\ell \in [L]$ and $k \in [K]$, sample $\sigma_i, \sigma_{i,k}, r, r_\ell \xleftarrow{\$} \mathbb{Z}_q$ conditioned on $\sum_{i\in[H]} \sigma_i = R$ and $\sum_{i\in[H]} \sigma_{k,i} = R_k$.*
*We consider the following oracles:*

$\underline{\tilde{\mathcal{O}}_{\mathbf{u}}}$**:** *On input $(\ell, i, \mathbf{x}^{(\mathsf{rep})}_{\ell,i}, \mathbf{x}'^{(\mathsf{rep})}_{\ell,i}) \in [L] \times [H] \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$, where $\mathsf{rep} \in [J_i]$ is a counter for the number of queries of the form $(\ell, i, \star, \star)$, sample $\rho^{(\mathsf{rep})}_{\ell,i} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{u}^{(\mathsf{rep})}_{\ell,i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}'^{(\mathsf{rep})}_{\ell,i},\ r_\ell,\ 0,\ \rho^{(\mathsf{rep})}_{\ell,i},\ 0^{2N+1})_{\mathbf{B}_i}\ .$$

$\underline{\boxed{\mathcal{O}^b_{\mathbf{u}}}}$**:** *For $b \in \{0,1\}$, on input $(i, \mathbf{x}^{(\tilde{j}_i)}_i) \in [H] \times \mathbb{Z}_q^N$, where $\tilde{j}_i \in [\tilde{J}_i]$ is a counter for the number of queries of the form $(i, \star)$, sample $\rho^{(\tilde{j}_i)}_i \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\text{If } \boxed{b=0}: \quad \mathbf{u}^{(\tilde{j}_i)}_i = (\boxed{\mathbf{x}^{(\tilde{j}_i)}_i},\ \boxed{0^N},\ r,\ 0,\ \rho^{(\tilde{j}_i)}_i,\ 0^{2N+1})_{\mathbf{B}_i}$$
$$\text{If } \boxed{b=1}: \quad \mathbf{u}^{(\tilde{j}_i)}_i = (\boxed{0^N},\ \boxed{\mathbf{x}^{(\tilde{j}_i)}_i},\ r,\ 0,\ \rho^{(\tilde{j}_i)}_i,\ 0^{2N+1})_{\mathbf{B}_i}\ .$$

$\underline{\mathcal{O}_{\mathbf{v}}}$**:** *On input $(i, \mathbf{y}^{(1,j_i)}_i, \mathbf{y}^{(0,j_i)}_i) \in [H] \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$, where $j_i \in [J_i]$ is a counter for the number of queries of the form $(i, \star, \star)$, sample $\pi^{(j_i)}_i \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{v}^{(j)}_i = (\mathbf{y}^{(1,j_i)}_i,\ \mathbf{y}^{(0,j_i)}_i,\ \sigma_i,\ \pi^{(j_i)}_i,\ 0,\ 0^{2N+1})_{\mathbf{B}^*_i}\ .$$

$\underline{\tilde{\mathcal{O}}_{\mathbf{v}}}$**:** *On inputs* $(k, i, \mathbf{y}_{k,i}^{(\mathsf{rep})}) \in [K] \times [H] \times \mathbb{Z}_q$, *where* $\mathsf{rep} \in [J_i]$ *is a counter for the number of queries of the form* $(k, i, \star)$, *sample* $\pi_{k,i}^{(\mathsf{rep})} \xleftarrow{\$} \mathbb{Z}_q$ *and output*

$$\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{2N+1})_{\mathbf{B}_i^*}\ .$$

*If* $\sum_{i=1}^{H} \langle \mathbf{x}_i^{(\tilde{j}_i)}, \mathbf{y}_i^{(0,j_i)} \rangle = \sum_{i=1}^{H} \langle \mathbf{x}_i^{(\tilde{j}_i)}, \mathbf{y}_i^{(1,j_i)} \rangle$ *for all* $\tilde{j}_i \in [\tilde{J}_i], j_i \in [J_i]$, *then the following advantage is negligible under the* SXDH *assumption:*

$$\begin{aligned}\Big| &\Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}},\boxed{\mathcal{O}_{\mathbf{u}}^0}}_{\tilde{\mathcal{O}}_{\mathbf{v}},\mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \tilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1] \\ &\quad - \Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}},\boxed{\mathcal{O}_{\mathbf{u}}^1}}_{\tilde{\mathcal{O}}_{\mathbf{v}},\mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \tilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1]\Big| \\ &\le (2N+8) \cdot \tilde{J} \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)\end{aligned}$$

*where* $\tilde{J} = \max_{i\in[H]} \tilde{J}_i$ *and* $\mathcal{A}$ *can query the oracles* $\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}_{\mathbf{u}}^b}, \mathcal{O}_{\mathbf{v}}, \tilde{\mathcal{O}}_{\mathbf{v}}$ adaptively, i.e. *the queries can be made in any order and any number of times respecting the (polynomial) upper bounds* $K, L, (J_i, \tilde{J}_i)_{i\in[H]}$.

## C.3 Proof of Security

**Security** The security of our scheme in Fig. 7 is proven below.

**Theorem 32.** *The DMCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *in Fig. 7 for the function class* $\mathcal{F}^{\mathsf{ip}}$ *is* one-challenge, weakly function-hiding *secure against* complete *queries under* static *corruption in the ROM, if the* SXDH *assumption holds for* $(\mathbb{G}_1, \mathbb{G}_2)$.
*More specifically, we let* $q_e$ *and* $q_k$ *denote the maximum number of distinct tags queried to* $\mathcal{O}\mathsf{Enc}$ *and* $\mathcal{O}\mathsf{KeyGen}$, *respectively. Furthermore, for* $i \in [n]$ *and* $\mathsf{tag}, \mathsf{tag\text{-}f} \in \mathsf{Tag}$, *we define* $J_{i,\mathsf{tag}}$ *and* $\tilde{J}_{i,\mathsf{tag\text{-}f}}$ *to be the numbers of queries of the form* $\mathcal{O}\mathsf{Enc}(i, \mathsf{tag}, \star, \star)$ *and* $\mathcal{O}\mathsf{KeyGen}(i, \mathsf{tag\text{-}f}, \star, \star)$, *and we set* $J \coloneqq \max_{i\in[n],\mathsf{tag}\in\mathsf{Tag}} J_{i,\mathsf{tag}}$ *and* $\tilde{J} \coloneqq \max_{i\in[n],\mathsf{tag\text{-}f}\in\mathsf{Tag}} \tilde{J}_{i,\mathsf{tag\text{-}f}}$, *respectively. Then, for any* ppt *adversary* $\mathcal{A}$ *against* $\mathcal{E}$, *we have the following bound:*

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

*Proof.* The proof is done via a sequence of hybrid games. The games are depicted in Figure 8.

**Game $\mathsf{G}_0$:** This is the experiment $\mathbf{Exp}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}fh}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ 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 $\mathsf{tag}^*$, the challenge function tag $\mathsf{tag\text{-}f}^*$ as well as the set $\mathcal{C} \subset [n]$ of corrupted clients. We define $\mathcal{H} \coloneqq [n] \setminus \mathcal{C}$. Knowing $\mathsf{tag}^*, \mathsf{tag\text{-}f}^*$, we index by $\ell \in [q_e]$ the $\ell$-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 $i \in [n]$, over all queried tags, 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

<u>**Game $\mathsf{G}_0$**</u>: $\sum_{i=1}^{n} \tilde{s}_i = \sum_{i=1}^{n} \tilde{t}_i = 0$, $\mathsf{H}_1(\mathsf{tag}_\ell) \to \llbracket \omega_\ell \rrbracket_1$, $\mathsf{H}_1(\mathsf{tag}^*) \to \llbracket \omega \rrbracket_1$, $\mathsf{H}_2(\mathsf{tag\text{-}f}_k) \to \llbracket \mu_k \rrbracket_2$, $\mathsf{H}_2(\mathsf{tag\text{-}f}^*) \to \llbracket \mu \rrbracket_2$, $b \xleftarrow{\$} \{0,1\}$ is the challenge bit

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

<u>**Game $\mathsf{G}_1$**</u>: $\sum_{i=1}^{n} s_{k,i} = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} t_{\ell,i} = \sum_{i=1}^{n} t_i = 0$ (Randomization)

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

<u>**Game $\mathsf{G}_2$**</u>: (Subspace Indistinguishability)

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

<u>**Game $\mathsf{G}_3$**</u>: (Swapping - Lemma 34 - over $\mathcal{O}\mathsf{Enc}$ tags)

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

<u>**Game $\mathsf{G}_4$**</u>: (Subspace Indistinguishability)

$$\begin{array}{rcc|c|c|c|c|c|cl}
\mathbf{d}_{k,i}^{(\tilde{j}')} & = ( & \boxed{0^N} & s_{k,i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \mathbf{y}_{k,i} & 0^{2N+1} & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{i}^{(\tilde{j})} & = ( & \boxed{0^N} & s_i & \mu & \pi_{i}^{(\tilde{j})} & 0 & \mathbf{y}_{i}^{(1,\tilde{j})} & 0^{2N+1} & )_{\mathbf{B}_i^*}
\end{array}$$

<u>**Game $\mathsf{G}_5$**</u>: (Subspace Indistinguishability)

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

<u>**Game $\mathsf{G}_6$**</u>: (Swapping - Lemma 34 - over $\mathcal{O}\mathsf{KeyGen}$ tags)

$$\begin{array}{rcc|c|c|c|c|c|cl}
\mathbf{d}_{k,i}^{(\tilde{j}')} & = ( & \boxed{\mathbf{y}_{k,i}} & s_{k,i} & \mu_k & \pi_{k,i}^{(\tilde{j}')} & 0 & \boxed{0^N} & 0^{2N+1} & )_{\mathbf{B}_i^*} \\
\mathbf{d}_{i}^{(\tilde{j})} & = ( & \boxed{\mathbf{y}_{i}^{(1,\tilde{j})}} & s_i & \mu & \pi_{i}^{(\tilde{j})} & 0 & \boxed{0^N} & 0^{2N+1} & )_{\mathbf{B}_i^*}
\end{array}$$

<u>**Game $\mathsf{G}_7$**</u>: (Cleaning - Subspace Indistinguishability)

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

Fig. 8: Games for proving Theorem 32

$s_{k,i} := \mu_k \cdot \tilde{s}_i$. Similarly, for the only challenge functional key query to $\mathsf{KeyGen}$ corresponding to $\mathsf{tag\text{-}f}^*$, we denote $\mathsf{H}_2(\mathsf{tag\text{-}f}^*) \to [\![\mu]\!]_2$ and define $s_i := \mu \cdot \tilde{s}_i$. We remark that for all $k \in [q_k]$, $(s_{k,i})_i$ is a secret sharing of 0, and the same holds for $(s_i)_i$ as well.
In the same manner, for the $\ell$-th non-challenge tag $\mathsf{tag}$, 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 $\mathsf{tag}^*$, 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 $b$

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

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 21 requires that $\mathbf{x}_i^{(0,j)} = \mathbf{x}_i^{(1,j)}$ (respectively $\mathbf{y}_i^0 = \mathbf{y}_i^1$) for all queries to $\mathcal{O}\mathsf{Enc}$ (respectively $\mathcal{O}\mathsf{DKeyGen}$) where $i \in \mathcal{C}$. All transitions, by means of *basis changes* in DPVS, in this proof 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 $\mathsf{G}_i$". 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 $\mathsf{G}_i$ versus that $\mathcal{A}$ outputs 1 in $\mathsf{G}_{i-1}$. We now start the description of games.

**Game $\mathsf{G}_1$:** The vectors are now:

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

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:

<u>$\mathsf{G}_{0.1}$</u>**:** We program $\mathsf{H}_1$ 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 $\mathsf{H}_2$. This gives a perfect simulation and $\Pr[\mathsf{G}_{0.1}] = \Pr[\mathsf{G}_0]$.

<u>$\mathsf{G}_{0.2}$</u>**:** 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 $\mathsf{DDH}$ 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 $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}$ under $\mathsf{tag}\text{-}\mathsf{f}_k$ as well as changing the challenge key query $\mathbf{d}_i^{(\tilde{\jmath})}$ undef $\mathsf{tag}\text{-}\mathsf{f}$.
In the hybrid $\mathsf{G}_{0.1.k}$ with $0 \le k \le q_k$, the first $k$ *non-challenge* key queries $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}$ are having a random secret shares over $i \in \mathcal{H}$:

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

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

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

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* $\tilde{\jmath}$ at position $i$ under $\mathsf{tag}\text{-}\mathsf{f}^*$, or $\tilde{\jmath}'$ under $\mathsf{tag}\text{-}\mathsf{f}_k$. 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 $\mathsf{DDH}$ instance $(\llbracket a \rrbracket_2, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2)$ where $c - ab = 0$ or a uniformly random value. Given a $\mathsf{ppt}$ 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 $\mathsf{ppt}$ adversary $\mathcal{B}$ that breaks the $\mathsf{DDH}$:

– The adversary $\mathcal{B}$ uses $\llbracket a \rrbracket_2$ to simulate $\mathcal{H}_2(\mathsf{tag}\text{-}\mathsf{f}_k)$ (or $\mathcal{H}_2(\mathsf{tag}\text{-}\mathsf{f}^*)$ if we are in the last transition to $\mathsf{G}_{0.1.q_k+1}$). This implicitly sets $\mu_k \coloneqq a$.
– The adversary $\mathcal{B}$ samples $\tilde{s}_i \xleftarrow{\$} \mathbb{Z}_q$ for corrupted $i$, as well as other parameters to output the corrupted keys $(\mathsf{ek}_i, \mathsf{sk}_i)$ to $\mathcal{A}$. Then, $\mathcal{B}$ computes and defines $S_k \coloneqq -\sum_{i\in\mathcal{C}} \tilde{s}_i$.
– Let us denote $H \coloneqq |\mathcal{H}|$ the number of honest $i$. For $i$ among the first $H-1$ honest clients whose keys are never leaked, $\mathcal{B}$ uses the *random-self reducibility* to compute $\llbracket \mu_k \tilde{s}_i \rrbracket_2$ for responding to the $k$-th key query $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}$ (or the challenge $\mathbf{d}_i^{(\tilde{\jmath})}$ if $k = q_k+1$).
– First of all, for $i$ among the first $|\mathcal{H}|-1$ honest, $\mathcal{B}$ samples $\alpha_{k,i}, \beta_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ and implicitly defines $b_{k,i} \coloneqq \alpha_{k,i} b + \beta_{k,i}$, $c_{k,i} \coloneqq \alpha_{k,i} c + \beta_{k,i} a$. We note that

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

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

$$\llbracket c_{k,H} \rrbracket_2 \coloneqq S_k \cdot \llbracket a \rrbracket_2 - \sum_{i\in\mathcal{H}\setminus\{H\}} \llbracket c_{k,i} \rrbracket_2 \tag{18}$$

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 $\mathsf{DDH}$ 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 $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}[N+1] = \mu_k \tilde{s}_i \coloneqq ab_{k,i}$ and we are in $\mathsf{G}_{0.1.k-1}$; Else $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}[N+1] = s_{k,i} \coloneqq c_{k,i}$ is a totally uniformly random value such that $\sum_{i\in\mathcal{H}} c_{k,i} + \mu_k \sum_{i\in\mathcal{C}} \tilde{s}_i = aS_k + \mu_k \sum_{i\in\mathcal{C}} \tilde{s}_i = 0$ thanks to (18) and the definition of $S_k$.

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

$\underline{\mathsf{G}_{0.3}}$**:** We replace the shifted shares $\omega\tilde{t}_i, \omega_\ell\tilde{t}_i$ in $\mathbf{c}_i^{(j)}, \mathbf{c}_{\ell,i}^{(j')}$ 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 $\mathsf{DDH}$ in $\mathbb{G}_1$ and with $(q_e + 2)$ hybrids (to change $q_e$ *non-challenge* ciphertext queries then the 1 *challenge* ciphertext). This gives us $|\Pr[\mathsf{G}_{0.3} = 1] - \Pr[\mathsf{G}_{0.2} = 1]| \le (q_e + 1) \cdot \mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

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

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

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

**Game $\mathsf{G}_2$:** We use $\mathsf{DSDH}$ in $\mathbb{G}_2$ to make $\mathbf{y}_{k,i}^{(\tilde{\jmath}')}$ appear in coordinates $[N+5, 2N+4]$ of $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}$, as well as $\mathbf{y}_i^{(1,\tilde{\jmath})}$ in coordinates $[N+5, 2N+4]$ of $\mathbf{d}_i^{(\tilde{\jmath})}$. This is of type *computational basis changes* that is reviewed in Appendix A.2, the calculation stays the same where we use $\mathsf{DSDH}$ to introduced *fixed* instead of random values.

We proceed by a sequence of $N+1$ hybrids, indexed by $m \in [0, N]$, such that the first hybrid of $m = 0$ is identical to $\mathsf{G}_1$ and for $m \ge 1$ in the $m$-th hybrid the first coordinates $[N+5, N+4+m]$ of $\mathbf{d}_{k,i}^{(\tilde{\jmath}')}, \mathbf{d}_i^{(\tilde{\jmath})}$ are modified. For $m \in [N]$, the transition from the $(m-1)$-th hybrid to the $m$-th hybrid is described below. Given a $\mathsf{DSDH}$ instance $([\![a]\!]_2, [\![b]\!]_2, [\![c]\!]_2)$ in $\mathbb{G}_2$ where $\delta \coloneqq c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

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

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

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

We update $(\pi_{k,i}^{(\tilde{\jmath}')}, \pi_i^{(\tilde{\jmath})})$ to $(\pi_{k,i}^{(\tilde{\jmath}')} + b\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m], \pi_i^{(\tilde{\jmath})} + b\mathbf{y}_i^{(1,\tilde{\jmath})}[m])$. Even though $\mathbf{b}_{i,N+1+m}$ 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:

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

If $\delta = 0$ we are in the $(m-1)$-th hybrid, else we are in the $m$-th hybrid. Totally, we proceed for all $i \in \mathcal{H}$ in parallel, after $N$ transitions we arrive at $\mathsf{G}_3$ and obtain $|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq 2N \cdot \mathbf{Adv}_{\mathbb{G}_2}^{\mathsf{DDH}}(1^\lambda)$.

**Game** $\mathsf{G}_3$**:** After $\mathsf{G}_2$ the vectors are now:

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

We now swap $\mathbf{x}_i^{(b,j)}, \mathbf{x}_{\ell,i}^{(j')}$ from coordinates $[1, N]$ to coordinates $[N+5, 2N+4]$ 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 $\mathsf{tag}_\ell$ to $\mathcal{O}\mathsf{Enc}$ and the only challenge tag $\mathsf{tag}^*$ that is declared at the beginning of the one-challenge game. The first hybrid is the same as $\mathsf{G}_3$. The transition between each hybrid is done by an application of Lemma 34. 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 $\omega_\ell$, *e.g.* their order of appearances. We verify the constraints required by Lemma 34.

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

- First of all, thanks to the *weakly function-hiding* admissibility (condition 3 that is concretely interpreted for inner products): for all $j \in [J], \tilde{j} \in [\tilde{J}]$

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

  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. This provides the conditions for the application of Lemma 34.
- The sets of vectors, listed in the order of the lemma's oracles, 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 constants will be $R = \sum_{i\in\mathcal{H}} s_i$, $R_k = \sum_{i\in\mathcal{H}} s_{k,i}$ for $k \in [q_k]$, known thanks to the *static* corruption. The $4N+4$ coordinates affected, in the order w.r.t the statement of Lemma 34 so that they form a subspace of dimension $4N+4$, are $([1,N], [N+5, 2N+4], N+1, N+3, N+4, [2N+5, 4N+5])$.

In the end, the security loss for this swapping is bounded by $(2N+8)\cdot J \cdot \mathbf{Adv}_{\mathbb{G}_1,\mathbb{G}_2}^{\mathsf{SXDH}}(1^\lambda)$.

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

- First of all, thanks to the *weakly function-hiding* admissibility (condition 3 that is concretely interpreted for inner products): 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. This provides the conditions for the application of Lemma 34.
- The sets of vectors, listed in the order of the lemma's oracles, 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 constants will be $R = \sum_{i\in\mathcal{H}} s_i$, $R_k = \sum_{i\in\mathcal{H}} s_{k,i}$ for $k \in [q_k]$, known thanks to the *static* corruption. The $4N+4$ coordinates affected, in the order w.r.t the statement of Lemma 34 so that they form a subspace of dimension $4N+4$, are $([1,N], [N+5, 2N+4], N+1, N+3, N+4, [2N+5, 4N+5])$.

Finally, the security loss for each swap over the $q_e$ non-challenge tags $\mathsf{tag}_\ell \neq \mathsf{tag}^*$ to $\mathcal{O}\mathsf{Enc}$ is upper bounded by: $(2N+8)\cdot J \cdot \mathbf{Adv}_{\mathbb{G}_1,\mathbb{G}_2}^{\mathsf{SXDH}}(1^\lambda)$. In total, we have $|\Pr[\mathsf{G}_3 = 1] - \Pr[\mathsf{G}_2 = 1]| \le (q_e+1)\cdot(2N+8)\cdot J \cdot \mathbf{Adv}_{\mathbb{G}_1,\mathbb{G}_2}^{\mathsf{SXDH}}(1^\lambda)$ in which $N$ is recalled to be the length of vectors encrypted by clients.

**Game $\mathsf{G}_4$:** After $\mathsf{G}_3$ the vectors are now:

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

In this $\mathsf{G}_4$, we use $\mathsf{DSDH}$ to clean $\mathbf{y}_{k,i}^{(\tilde{j}')}$ in coordinates $[1,N]$ of $\mathbf{d}_{k,i}^{(\tilde{j}')}$, as well as $\mathbf{y}_i^{(b,\tilde{j})}$ in coordinates $[1,N]$ of $\mathbf{d}_i^{(\tilde{j})}$.

We again exploit the randomness at coordinate $(N+3)$ of the $\mathbf{d}$-vectors and proceed by a sequence of $N+1$ hybrids, indexed by $m \in [0,N]$, such that the first hybrid for $m=0$ is identical to $\mathsf{G}_3$ while in the $m$-th hybrid the first coordinates $[1,m]$ of $\mathbf{d}^{(\tilde{\jmath}')}, \mathbf{d}_i^{(\tilde{\jmath})}$ are modified, for $m \geq 1$. For $m \in [N]$, the transition from the $(m-1)$-th hybrid to the $m$-th hybrid can be done by a *computational basis change* using a $\mathsf{DSDH}$ instance $(\llbracket a \rrbracket_2, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2)$ in $\mathbb{G}_2$ where $\delta := c - ab$ is either 0 or 1. The bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

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

The basis $\mathbf{B}_i^*$ can be computed using $\llbracket a \rrbracket_2$ and the $\mathbf{d}$-vectors are simulated below:

$$\begin{aligned}
\mathbf{d}_{k,i}^{(\tilde{\jmath}')} &= (\underbrace{0,..,0,\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m],..,\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[N]}_{\text{first } (m-1)\text{-th coords are } 0},\ s_{k,i},\ \mu_k,\ \pi_{k,i}^{(\tilde{\jmath}')},\ 0,\ \mathbf{y}_{k,i}^{(\tilde{\jmath}')},\ 0^{2N+1})_{\mathbf{B}_i^*} \\
&\quad + (\underbrace{0,..,0,-c\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m],0,..,0}_{m\text{-th coord among } N},\ 0^2,\ b\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m],\ 0,\ 0^{3N+1})_{\mathbf{H}_i^*} \\
&= (\underbrace{0,..,0,\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m] - \delta\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m],..,\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[N]}_{\text{first } (m-1)\text{-th coords are } 0},\ s_{k,i},\ \mu_k,\ \pi_{k,i}^{(\tilde{\jmath}')} + b\mathbf{y}_{k,i}^{(\tilde{\jmath}')}[m],\ 0,\ \mathbf{y}_{k,i}^{(\tilde{\jmath}')},\ 0^{2N+1})_{\mathbf{B}_i^*} \\
\mathbf{d}_i^{(\tilde{\jmath})} &= (\underbrace{0,..,0,\mathbf{y}_i^{(b,\tilde{\jmath})}[m],..,\mathbf{y}_i^{(b,\tilde{\jmath})}[N]}_{\text{first } (m-1)\text{-th coords are } 0},\ s_i,\ \mu,\ \pi_i^{(\tilde{\jmath})},\ 0,\ \mathbf{y}_i^{(1,\tilde{\jmath})},\ 0^{2N+1})_{\mathbf{B}_i^*} \\
&\quad + (\underbrace{0,..,0,-c\mathbf{y}_i^{(b,\tilde{\jmath})}[m],0,..,0}_{m\text{-th coord among } N},\ 0^2,\ b\mathbf{y}_i^{(b,\tilde{\jmath})}[m],\ 0,\ 0^{3N+1})_{\mathbf{H}_i^*} \\
&= (\underbrace{0,..,0,\mathbf{y}_i^{(b,\tilde{\jmath})}[m] - \delta\mathbf{y}_i^{(b,\tilde{\jmath})}[m],..,\mathbf{y}_i^{(b,\tilde{\jmath})}[N]}_{\text{first } (m-1)\text{-th coords are } 0},\ s_{k,i},\ \mu_k,\ \pi_{k,i}^{(\tilde{\jmath}')} + b\mathbf{y}_i^{(b,\tilde{\jmath})}[m],\ 0,\ \mathbf{y}_i^{(1,\tilde{\jmath})},\ 0^{2N+1})_{\mathbf{B}_i^*} \ .
\end{aligned}$$

Even though we cannot compute $\mathbf{b}_{i,m}$ due to the lack of $\llbracket a \rrbracket_1$, the $\mathbf{c}$-vectors can be written directly in $\mathbf{H}_i^*$:

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

If $\delta = 0$ we are in the $(m-1)$-th hybrid, else we are in the $m$-th hybrid. Totally, after $N$ transitions we arrive at $\mathsf{G}_4$ and obtain $|\Pr[\mathsf{G}_4 = 1] - \Pr[\mathsf{G}_3 = 1]| \leq 2N \cdot \mathbf{Adv}_{\mathbb{G}_2}^{\mathsf{DDH}}(1^\lambda)$.

**Game** $\mathsf{G}_5$**:** After $\mathsf{G}_4$ the vectors are now:

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

We use $\mathsf{DSDH}$ in $\mathbb{G}_1$ to make $\mathbf{x}_{\ell,i}^{(j')}$ appear in coordinates $[1,N]$ of $\mathbf{c}_{\ell,i}^{(j')}$, as well as $\mathbf{x}_i^{(1,j)}$ in coordinates $[1,N]$ of $\mathbf{c}_i^{(j)}$. This is of type *computational basis changes* that is reviewed in

Appendix A.2, the calculation stays the same where we use $\mathsf{DSDH}$ to introduced *fixed* instead of random values.
We exploit the randomness at coordinate $(N+4)$ of the $\mathbf{c}$-vectors and proceed by a sequence of $N+1$ hybrids, indexed by $m \in [0, N]$, such that the first hybrid for $m = 0$ is identical to $\mathsf{G}_4$ while in the $m$-th hybrid the first coordinates $[1, m]$ of $\mathbf{c}_{\ell,i}^{(j')}, \mathbf{c}_i^{(j)}$ are modified, for $m \geq 1$. For $m \in [N]$, the transition from the $(m-1)$-th hybrid to the $m$-th hybrid can be done by a *computational basis change* using a $\mathsf{DSDH}$ instance $(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket c \rrbracket_1)$ in $\mathbb{G}_1$ where $\delta := c - ab$ is either 0 or 1. The bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

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

The calculation can be adapted from that in the transitions from $\mathsf{G}_1$ to $\mathsf{G}_2$, except that now we will do it *dually* for the $\mathbf{c}$-vectors in the basis change from $\mathbf{H}_i$ to $\mathbf{B}_i$. The $\mathbf{B}_i$ can be computed using $\llbracket a \rrbracket_1$. Even though we cannot compute $\mathbf{b}_{i,m}^*$ due to the lack of $\llbracket a \rrbracket_2$, the $\mathbf{d}$-vectors can be written directly in $\mathbf{H}_i^*$ and stay invariant thanks to the fact that their coordinates $[1, N]$ are all 0 after $\mathsf{G}_4$. Totally, after $N$ transitions we arrive at $\mathsf{G}_5$ and obtain $|\Pr[\mathsf{G}_5 = 1] - \Pr[\mathsf{G}_4 = 1]| \leq 2N \cdot \mathbf{Adv}_{\mathbb{G}_1}^{\mathsf{DDH}}(1^\lambda)$.

**Game $\mathsf{G}_6$:** After $\mathsf{G}_5$ the vectors are now:

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

We apply Lemma 34 to swap $\mathbf{y}_i^{(1,\tilde{j})}, \mathbf{y}_{k,i}^{(\tilde{j}')}$ from coordinates $[N+5, 2N+4]$ to coordinates $[1, N]$ 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 $\mathsf{tag}\text{-}\mathsf{f}_k$ to $\mathcal{O}\mathsf{KeyGen}$ and the only challenge tag $\mathsf{tag}\text{-}\mathsf{f}$ that is declared at the beginning of the one-challenge game. The first hybrid is the same as $\mathsf{G}_5$. The transition between each hybrid is done by an application of Lemma 34. 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 $\mu_k$, *e.g.* their order of appearances. We verify the constraints required by Lemma 34.

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

- First of all, thanks to the *weakly function-hiding* admissibility (condition 3 that is concretely interpreted for inner products): for all $j \in [J], \tilde{j} \in [\tilde{J}]$

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

  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. This provides the conditions for the application of Lemma 34.
- The sets of vectors, listed in the order of the lemma's oracles, 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 constants will be $R = \sum_{i \in \mathcal{H}} t_i$, $R_\ell = \sum_{i \in \mathcal{H}} \mu_{\ell,i}$ for $\ell \in [q_e]$, known thanks to the *static* corruption. The $4N + 4$ coordinates affected, in the order w.r.t the statement of Lemma 34 so that they form a subspace of dimension $4N + 4$, are $([N+5, 2N+4], [1, N], N+2, N+4, N+3, [2N+5, 4N+5])$.

Finally this swap incurs a security loss upper bounded by $(2N+8)\cdot\tilde{J}\cdot\mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$.

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

- First of all, thanks to the *weakly function-hiding* admissibility (condition 3 that is concretely interpreted for inner products): for all $j\in[J],\tilde{j}'\in[\tilde{J}]$

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

  where (1) comes from condition 1 that for corrupted $i$ the challenge messages satisfy $\mathbf{x}^{(b,j)}_i=\mathbf{x}^{(1,j)}_i$, and (2) comes from the weakly function-hiding. This provides the conditions for the application of Lemma 34.
- The sets of vectors, listed in the order of the lemma's oracles, are $((\mathbf{d}^{(\tilde{j})}_i)^{\tilde{j}\in[J]}_{i\in\mathcal{H}},(\mathbf{d}^{(\tilde{j}')}_{k,i})^{\tilde{j}'\in[J]}_{i\in\mathcal{H},k\in[q_k]},(\mathbf{c}^{(j)}_i)^{j\in[J]}_{i\in\mathcal{H}},(\mathbf{c}^{(j')}_{\ell,i})^{j'\in[J]}_{i\in\mathcal{H}})$. The constants will be $R=\sum_{i\in\mathcal{H}}t_i$, $R_\ell=\sum_{i\in\mathcal{H}}\mu_{\ell,i}$ for $\ell\in[q_e]$, known thanks to the *static* corruption. The $4N+4$ coordinates affected, in the order w.r.t the statement of Lemma 34 so that they form a subspace of dimension $4N+4$, are $([N+5,2N+4],[1,N],N+2,N+4,N+3,[2N+5,4N+5])$.

Finally, the security loss for each swap over the $q_e$ non-challenge tags to $\mathcal{O}\mathsf{Enc}$ is upper bounded by: $(2N+8)\cdot\tilde{J}\cdot\mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$. In total, we have $|\Pr[\mathsf{G}_6=1]-\Pr[\mathsf{G}_5=1]|\le(q_k+1)\cdot(2N+8)\cdot\tilde{J}\cdot\mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$ in which $N$ is recalled to be the length of vectors encrypted by clients.

**Game $\mathsf{G}_7$:** After $\mathsf{G}_6$ the vectors are now:

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

We perform some cleanings to make the vectors independent of $b$. We use $\mathsf{DSDH}$ in $\mathbb{G}_1$ to clean $\mathbf{x}^{(j')}_{\ell,i}$ in coordinates $[N+5,2N+4]$ of $\mathbf{c}^{(j')}_{\ell,i}$, as well as $\mathbf{x}^{(b,j)}_i$ in coordinates $[N+5,2N+4]$ of $\mathbf{c}^{(j)}_i$. This is of type *computational basis changes* that is reviewed in Appendix A.2, the calculation stays the same where we use $\mathsf{DSDH}$ to introduced *fixed* instead of random values.

We exploit the randomness at coordinate $(N+4)$ of the $\mathbf{c}$-vectors and proceed by a sequence of $N+1$ hybrids, indexed by $m\in[0,N]$, such that the first hybrid for $m=0$ is identical to $\mathsf{G}_6$ while in the $m$-th hybrid the coordinates $[N+5,N+4+m]$ of $\mathbf{c}^{(j')}_{\ell,i},\mathbf{c}^{(j)}_i$ are modified, for $m\ge1$. For $m\in[N]$, the transition from the $(m-1)$-th hybrid to the $m$-th hybrid can be done by a *computational basis change* using a $\mathsf{DSDH}$ 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:

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

The calculation can be adapted from that in the transitions from $\mathsf{G}_3$ to $\mathsf{G}_4$, except that now we will do the cleaning *dually* for the $\mathbf{c}$-vectors w.r.t the basis change from $\mathbf{H}_i$ to $\mathbf{B}_i$. The $\mathbf{B}_i$ can be computed using $[\![a]\!]_1$. Even though we cannot compute $\mathbf{b}^*_{i,m}$ due to the lack of $[\![a]\!]_2$, the $\mathbf{d}$-vectors can be written directly in $\mathbf{H}^*_i$ and stay invariant thanks to the fact that their coordinates $[N+5,2N+4]$ are all 0 after $\mathsf{G}_6$. Totally, after $N$ transitions we arrive at $\mathsf{G}_7$ and obtain $|\Pr[\mathsf{G}_7=1]-\Pr[\mathsf{G}_6=1]|\le2N\cdot\mathbf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(1^\lambda)$.

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

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

and they do not depend on the challenge bit $b \xleftarrow{\$} \{0,1\}$ anymore and $\Pr[\mathsf{G}_7 = 1] = 1/2$. The difference in advantages is

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{1chal\text{-}pos\text{-}stat\text{-}wfh}}_{\mathcal{E},\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n},\mathcal{A}}(1^\lambda) &= |\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}|\\
&= |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_7 = 1]|\\
&\le \sum_{i=1}^{7} |\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i-1} = 1]|\\
&\le \Big(\big((q_e+1)J + (q_k+1)\tilde{J}\big)\cdot(2N+8) + q_k + q_e + 8N + 2\Big)\cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)
\end{aligned}$$

and the proof is completed. □

## C.4 Technical Overview of the Swapping Lemma 34

**Lemma 34 (Swapping).** *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda), J_i = J_i(\lambda), \widetilde{J}_i = \widetilde{J}_i(\lambda), N = N(\lambda) \in \mathbb{N}$ where $i \in [H]$ and $H, K, L, J_i, \widetilde{J}_i, N : \mathbb{N} \to \mathbb{N}$ are polynomials. Let $(\mathbf{B}_i, \mathbf{B}_i^*)$, for each $i \in [H]$, be a pair of random dual bases of dimension $4N+4$ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$. All basis vectors are kept secret. Let $R, R_1, \ldots, R_K \in \mathbb{Z}_q$ be some public scalars. For $i \in [H]$, $\ell \in [L]$ and $k \in [K]$, sample $\sigma_i, \sigma_{i,k}, r, r_\ell \xleftarrow{\$} \mathbb{Z}_q$ conditioned on $\sum_{i\in[H]} \sigma_i = R$ and $\sum_{i\in[H]} \sigma_{k,i} = R_k$.*
*We consider the following oracles:*

$\underline{\tilde{\mathcal{O}}_\mathbf{u}}$**:** *On input $(\ell, i, \mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x}'^{(\mathsf{rep})}_{\ell,i}) \in [L]\times[H]\times\mathbb{Z}_q^N\times\mathbb{Z}_q^N$, where $\mathsf{rep} \in [J_i]$ is a counter for the number of queries of the form $(\ell, i, \star, \star)$, sample $\rho_{\ell,i}^{(\mathsf{rep})} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x}'^{(\mathsf{rep})}_{\ell,i},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{2N+1})_{\mathbf{B}_i}\ .$$

$\underline{\boxed{\mathcal{O}_\mathbf{u}^b}}$**:** *For $b \in \{0,1\}$, on input $(i, \mathbf{x}_i^{(\tilde{j}_i)}) \in [H]\times\mathbb{Z}_q^N$, where $\tilde{j}_i \in [\tilde{J}_i]$ is a counter for the number of queries of the form $(i, \star)$, sample $\rho_i^{(\tilde{j}_i)} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\begin{aligned}
\text{If } \boxed{b=0}:&\quad \mathbf{u}_i^{(\tilde{j}_i)} = (\boxed{\mathbf{x}_i^{(\tilde{j}_i)}},\ \boxed{0^N},\ r,\ 0,\ \rho_i^{(\tilde{j}_i)},\ 0^{2N+1})_{\mathbf{B}_i}\\
\text{If } \boxed{b=1}:&\quad \mathbf{u}_i^{(\tilde{j}_i)} = (\boxed{0^N},\ \boxed{\mathbf{x}_i^{(\tilde{j}_i)}},\ r,\ 0,\ \rho_i^{(\tilde{j}_i)},\ 0^{2N+1})_{\mathbf{B}_i}\ .
\end{aligned}$$

$\underline{\mathcal{O}_\mathbf{v}}$**:** *On input $(i, \mathbf{y}_i^{(1,j_i)}, \mathbf{y}_i^{(0,j_i)}) \in [H]\times\mathbb{Z}_q^N\times\mathbb{Z}_q^N$, where $j_i \in [J_i]$ is a counter for the number of queries of the form $(i, \star, \star)$, sample $\pi_i^{(j_i)} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j_i)},\ \mathbf{y}_i^{(0,j_i)},\ \sigma_i,\ \pi_i^{(j_i)},\ 0,\ 0^{2N+1})_{\mathbf{B}_i^*}\ .$$

$\underline{\tilde{\mathcal{O}}_\mathbf{v}}$**:** *On inputs $(k, i, \mathbf{y}_{k,i}^{(\mathsf{rep})}) \in [K]\times[H]\times\mathbb{Z}_q$, where $\mathsf{rep} \in [J_i]$ is a counter for the number of queries of the form $(k, i, \star)$, sample $\pi_{k,i}^{(\mathsf{rep})} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{2N+1})_{\mathbf{B}_i^*}\ .$$

*If* $\sum_{i=1}^{H}\langle \mathbf{x}_i^{(\tilde{j}_i)}, \mathbf{y}_i^{(0,j_i)}\rangle = \sum_{i=1}^{H}\langle \mathbf{x}_i^{(\tilde{j}_i)}, \mathbf{y}_i^{(1,j_i)}\rangle$ *for all* $\tilde{j}_i \in [\widetilde{J}_i], j_i \in [J_i]$, *then the following advantage is negligible under the* SXDH *assumption:*

$$\begin{aligned}\Bigg| &\Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}^0_{\mathbf{u}}}}_{\tilde{\mathcal{O}}_{\mathbf{v}}, \mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \widetilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1] \\ &\quad - \Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}^1_{\mathbf{u}}}}_{\tilde{\mathcal{O}}_{\mathbf{v}}, \mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \widetilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1]\Bigg| \\ &\leq (2N+8)\cdot \tilde{J} \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)\end{aligned}$$

*where* $\tilde{J} = \max_{i\in[H]} \tilde{J}_i$ *and* $\mathcal{A}$ *can query the oracles* $\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}^b_{\mathbf{u}}}, \mathcal{O}_{\mathbf{v}}, \tilde{\mathcal{O}}_{\mathbf{v}}$ adaptively, i.e. *the queries can be made in any order and any number of times respecting the (polynomial) upper bounds* $K, L, (J_i, \widetilde{J}_i)_{i\in[H]}$.

The proof is done via a sequence of hybrids over the repetitions $\tilde{j} \in [\tilde{J}]$ where we separate the repetition $\mathbf{u}_i^{(\tilde{j})}$ (if $\tilde{J}_i < \tilde{j}$ there is no change on $\mathbf{u}_i$) into isolated coordinates then apply a lemma that treats the fundamental case where $\widetilde{J} = 1$ (Lemma 35). A full proof of the general Lemma 34 can be found in Appendix C.6. Below we give the main ideas for the simpler case $\widetilde{J} = 1$.

*Proof (Main ideas for Lemma 35).* The sequence of games is given in Figure 9. We explain the main steps in our proof as follows, where details about *formal* and *computational* basis changes can be revised from the examples in **Basis changes** of Appendix A.2. 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 $\boxed{\text{boxes}}$. We omit the index of repetitions over $(\mathbf{u}_i)_i$, because $\widetilde{J} = 1$, for the ease of presentation. For each $i \in [H]$, the value $J_i$ denotes the maximum number of possible repetitions $(\mathbf{u}_{\ell,i}^{(\mathsf{rep})})_{\mathsf{rep}}$, $(\mathbf{v}_i^{(j)})_j$, and $(\mathbf{v}_{k,i}^{(\mathsf{rep})})_{\mathsf{rep}}$, indexed by $\mathsf{rep}$ and $j$ over all $\ell, k$.
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 $\sigma_i$ 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 $(\mathbf{v}_i^{(j)}[4N_i+4])_{i=1}^H$.

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

We use the hypothesis that all basis vectors are kept secret so that the computational basis change using DDH cannot be detected by the adversary. More details can be found in the transition $\mathsf{G}_0 \to \mathsf{G}_1$. After $\mathsf{G}_1$, we perform a *formal* duplication to go to $\mathsf{G}_2$ 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 $i \in [H], k \in [K], j \in [J]$.

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

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

<u>**Game $\mathsf{G}_0$**</u>: The vectors are sampled according to $D_0$.

<u>**Game $\mathsf{G}_1$**</u>: (Random 0-Secret Sharing) $\forall\, j \in [J] : \sum_{i=1}^{H} \tau_i = 0$

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

<u>**Game $\mathsf{G}_2$**</u>: (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 $\mathbf{B}_i^*$)

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

<u>**Game $\mathsf{G}_3$**</u>: (Computational Swapping between $[1, N_i]$ and $[2N_i+4, 3N_i+3]$ in $\mathbf{u}_i$ using $(2N_i+3)$-randomness in $\mathbf{B}_i$)

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

Inside a *complexity leveraging* argument:

<u>**Game $\mathsf{G}_4$**</u>: (Formal Quotient on coordinates $[2N_i+4, 4N_i+3]$ in $\mathbf{B}_i$)

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

<u>**Game $\mathsf{G}_5$**</u>: $\Delta\mathbf{y}_i^{(j)} := \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}, \tilde{\tau}_i := \tau_i + \frac{1}{r'}\langle \mathbf{x}_i, \Delta\mathbf{y}_i^{(j)} \rangle$ (Formal Swapping)

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

<u>**Game $\mathsf{G}_6$**</u>: (Formal Quotient on coordinates $[2N_i+4, 4N_i+3]$ in $\mathbf{B}_i$)

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

<u>**Game $\mathsf{G}_7$**</u>: (Computational Swapping between $[1, N_i]$ and $[3N_i+4, 4N_i+3]$ in $\mathbf{u}_i$ using $(2N_i+3)$-randomness in $\mathbf{B}_i$)

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

<u>**Game $\mathsf{G}_8$**</u>: Undo $\mathsf{G}_2$, $\mathsf{G}_1$ (Cleaning) – Vectors sampled according to $D_1$.

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

Fig. 9: Games for proving Lemma 34 in the particular case where $\widetilde{J} = 1$. We omit the index of repetitions over $(\mathbf{u}_i)_i$ for the ease of presentation.

“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 C.5.
After $\mathsf{G}_2$, we perform a *computational* basis change under SXDH in order to swap between $[1, N_i]$ and $[2N_i+4, 3N_i+3]$ in $\mathbf{u}_i$. The randomness is taken from $\rho_i$ at coordinate $2N_i+3$ in $\mathbf{u}_i$.

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

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.
Upon reaching $\mathsf{G}_3$, we obtain the necessary ingredients for our proof. A *formal* basis change maintains identical views for the adversary in two games, allowing a *complexity leveraging* argument. This argument aims to demonstrate that the adversary's views over two hybrids are perfectly *identical*, resulting in a 0 difference in winning advantages under efficient simulation. Principally a complexity argument consists of a formal argument on top of two identical variants of hybrids, which are usually their *selective* variants, and yields a security loss of 0. Formal basis changes provide a way to link underlying selective variants that require identical views. The formal changes highlight DPVS's information-theoretic properties, as discussed in **Basis changes** of Appendix A.2. However, the primary obstacle is handling the modification of all vectors under basis changes.
We now explain the sequence of games on which the complexity leveraging is applied. 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 $\mathsf{G}_6$ whose vectors are:

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

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:

$$\Pr[\mathsf{G}_3^* = 1] = \Pr[\mathsf{G}_4^* = 1] = \Pr[\mathsf{G}_5^* = 1] = \Pr[\mathsf{G}_6^* = 1] \ . \tag{19}$$

We briefly highlight the selective 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 (22) 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 $\mathbf{u}_i[4N_i+4]$ (introduced from $\mathsf{G}_1$) 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 (23). We note that unlike $\mathbf{u}_i$, the vectors $\mathbf{u}_{\ell,i}^{(\mathsf{rep})}$ stay invariant because $\mathbf{u}_{\ell,i}^{(\mathsf{rep})}[4N_i+4] = 0$.

  - Dually, all $\mathbf{v}$-vectors will be altered such that the *accumulated differences* $\sum_{k=1}^{N_i}(\mathbf{v}_i^{(j)}[2N_i+3+k]-\mathbf{v}_i^{(j)}[3N_i+3+k])$ will be added to $\mathbf{v}_i^{(j)}[4N_i+4]$. 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 $\mathsf{rep}$. 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 $j\in[J]$, *i.e.* for a given $k\in[N_i]$, the difference $\mathbf{v}_i^{(j)}[2N_i+3+k]-\mathbf{v}_i^{(j)}[3N_i+3+k]=\mathbf{x}_i\varDelta\mathbf{y}_i^{(j)}[k]$, where $\varDelta\mathbf{y}_i^{(j)}[k]\coloneqq\mathbf{y}_i^{(1,j)}[k]-\mathbf{y}_i^{(0,j)}[k]$, can be non-zero.
  - We note that for each $i\in[H]$, for all $j\in[J]$, the term $\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j)}\rangle=\sum_{k\in[N_i]}\mathbf{x}_i\varDelta\mathbf{y}_i^{(j)}[k]$ is a constant. Otherwise there exists $\varnothing\neq I'\subseteq[H]$ and $j',j''\in[J]$ so that

$$\sum_{i\in I'}\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j')}\rangle\neq\sum_{i\in I'}\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j'')}\rangle$$

while

$$\sum_{i\in[H]\setminus I'}\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j')}\rangle=\sum_{i\in[H]\setminus I'}\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j'')}\rangle\ ,$$

which contradicts 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 any $j\in[J]$.
  - It is at this point that we need to use the secret sharing $(\tau_i)_{i=1}^H$ of 0 in $(\mathbf{v}_i^{(j)}[4N_i+4])_{i=1}^H$ (introduced from $\mathsf{G}_1$) as well as the above observation that for each $i\in[H]$, for all $j\in[J]$, the term $\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j)}\rangle$ is constant. More specifically, adding $\sum_{k=1}^{N_i}(\mathbf{v}_i^{(j)}[2N_i+3+k]-\mathbf{v}_i^{(j)}[3N_i+3+k])$ meaning adding a fixed multiple of $\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j)}\rangle$ to $\tau_i$, which is constant for whatever $j$. This fixed multiple of a constant $\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j)}\rangle$ over $j$ still keeps it a sharing of 0 over all $i\in[H]$ and the new $\tilde{\tau}_i$ still does not depend on $j$:

$$\sum_{i\in[H]}(\tau_i+\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j)}\rangle)=\sum_{i\in[H]}\tau_i+\sum_{i\in[H]}\langle\mathbf{x}_i,\varDelta\mathbf{y}_i^{(j)}\rangle=0\ .$$

This "fixed multiple" depends only on $r'\xleftarrow{\$}\mathbb{Z}_q^*$ (introduced from $\mathsf{G}_1$) 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 11) of the complexity leveraging makes use of the fact that the "good" event happens with a fixed probability in conjunction with property (19), 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 $\mathsf{G}_7\to\mathsf{G}_8$). $\square$

## C.5 Swapping without Repetitions – Proof of Lemma 34 (Special Case)

We prove a special case of Lemma 34 where $\widetilde{J}=1$. We omit the index of repetitions over $(\mathbf{u}_i)_i$ for the ease of presentation.

**Lemma 35 (Swapping without Repetitions).** *Let $\lambda\in\mathbb{N}$ and $H=H(\lambda),K=K(\lambda),L=L(\lambda),J_i=J_i(\lambda),N=N(\lambda)\in\mathbb{N}$ where $i\in[H]$ and $H,K,L,J_i,\widetilde{J}_i,N:\mathbb{N}\to\mathbb{N}$ are polynomials. Let $(\mathbf{B}_i,\mathbf{B}_i^*)$, for each $i\in[H]$, be a pair of random dual bases of dimension $4N+4$ in $(\mathbb{G}_1,\mathbb{G}_2,\mathbb{G}_{\mathrm{t}},g_1,g_2,g_{\mathrm{t}},\mathbf{e},q)$. All basis vectors are kept secret. Let $R,R_1,\ldots,R_K\in\mathbb{Z}_q$ be some public scalars. For $i\in[H]$, $\ell\in[L]$ and $k\in[K]$, sample $\sigma_i,\sigma_{i,k},r,r_\ell\xleftarrow{\$}\mathbb{Z}_q$ conditioned on $\sum_{i\in[H]}\sigma_i=R$ and $\sum_{i\in[H]}\sigma_{k,i}=R_k$.*
*We consider the following oracles:*

$\underline{\tilde{\mathcal{O}}_{\mathbf{u}}}$**:** *On input* $(\ell, i, \mathbf{x}_{\ell,i}^{(\mathsf{rep})}, \mathbf{x'}_{\ell,i}^{(\mathsf{rep})}) \in [L] \times [H] \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$*, where* $\mathsf{rep} \in [J_i]$ *is a counter for the number of queries of the form* $(\ell, i, \star, \star)$*, sample* $\rho_{\ell,i}^{(\mathsf{rep})} \xleftarrow{\$} \mathbb{Z}_q$ *and output*

$$\mathbf{u}_{\ell,i}^{(\mathsf{rep})} = (\mathbf{x}_{\ell,i}^{(\mathsf{rep})},\ \mathbf{x'}_{\ell,i}^{(\mathsf{rep})},\ r_\ell,\ 0,\ \rho_{\ell,i}^{(\mathsf{rep})},\ 0^{2N+1})_{\mathbf{B}_i}\ .$$

$\underline{\boxed{\mathcal{O}_{\mathbf{u}}^b}}$**:** *For* $b \in \{0,1\}$*, on input* $(i, \mathbf{x}_i) \in [H] \times \mathbb{Z}_q^N$*, sample* $\rho_i \xleftarrow{\$} \mathbb{Z}_q$ *and output*

$$\begin{aligned} \textit{If } \boxed{b=0}: \quad & \mathbf{u}_i = (\boxed{\mathbf{x}_i},\ \boxed{0^N},\ r,\ 0,\ \rho_i,\ 0^{2N+1})_{\mathbf{B}_i} \\ \textit{If } \boxed{b=1}: \quad & \mathbf{u}_i = (\boxed{0^N},\ \boxed{\mathbf{x}_i},\ r,\ 0,\ \rho_i,\ 0^{2N+1})_{\mathbf{B}_i}\ . \end{aligned}$$

*This oracle can be called only once for each* $i \in [H]$*.*

$\underline{\mathcal{O}_{\mathbf{v}}}$**:** *On input* $(i, \mathbf{y}_i^{(1,j_i)}, \mathbf{y}_i^{(0,j_i)}) \in [H] \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$*, where* $j_i \in [J_i]$ *is a counter for the number of queries of the form* $(i, \star, \star)$*, sample* $\pi_i^{(j_i)} \xleftarrow{\$} \mathbb{Z}_q$ *and output*

$$\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j_i)},\ \mathbf{y}_i^{(0,j_i)},\ \sigma_i,\ \pi_i^{(j_i)},\ 0,\ 0^{2N+1})_{\mathbf{B}_i^*}\ .$$

$\underline{\tilde{\mathcal{O}}_{\mathbf{v}}}$**:** *On inputs* $(k, i, \mathbf{y}_{k,i}^{(\mathsf{rep})}) \in [K] \times [H] \times \mathbb{Z}_q$*, where* $\mathsf{rep} \in [J_i]$ *is a counter for the number of queries of the form* $(k, i, \star)$*, sample* $\pi_{k,i}^{(\mathsf{rep})} \xleftarrow{\$} \mathbb{Z}_q$ *and output*

$$\mathbf{v}_{k,i}^{(\mathsf{rep})} = (\mathbf{y}_{k,i}^{(\mathsf{rep})},\ \mathbf{y}_{k,i}^{(\mathsf{rep})},\ \sigma_{k,i},\ \pi_{k,i}^{(\mathsf{rep})},\ 0,\ 0^{2N+1})_{\mathbf{B}_i^*}\ .$$

*If* $\sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(0,j_i)} \rangle = \sum_{i=1}^H \langle \mathbf{x}_i, \mathbf{y}_i^{(1,j_i)} \rangle$ *for all* $j_i \in [J_i]$*, then the following advantage is negligible under the* $\mathsf{SXDH}$ *assumption:*

$$\begin{aligned} &\Big| \Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}_{\mathbf{u}}^0}}_{\tilde{\mathcal{O}}_{\mathbf{v}}, \mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1] \\ &\quad - \Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}_{\mathbf{u}}^1}}_{\tilde{\mathcal{O}}_{\mathbf{v}}, \mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1] \Big| \\ &\leq (2N+8) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1, \mathbb{G}_2}(1^\lambda) \end{aligned}$$

*where* $\mathcal{A}$ *can query the oracles* $\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}_{\mathbf{u}}^b}, \mathcal{O}_{\mathbf{v}}, \tilde{\mathcal{O}}_{\mathbf{v}}$ adaptively, i.e. *the queries can be made in any order and any number of times respecting the (polynomial) upper bounds* $K, L, (J_i)_{i\in[H]}$*.*

*Proof (of Lemma 35).* The sequence of games is given in Figure 9. The changes that make the transition between games are highlighted by a $\boxed{\text{frame}}$. For the sake of simplicity, we do not mention explicitly the oracles for the generation of vectors in the proof. We write $0^t$ to denote $t$ consecutive coordinates containing 0. For each $i \in [H]$, the value $J_i$ denotes the maximum number of possible repetitions $(\mathbf{u}_{\ell,i}^{(\mathsf{rep})})_{\mathsf{rep}}$, $(\mathbf{v}_i^{(j)})_j$, and $(\mathbf{v}_{k,i}^{(\mathsf{rep})})_{\mathsf{rep}}$, indexed by $\mathsf{rep}$ and $j$ over all $\ell, k$. We define $J := \max i \in [H] J_i$. The details of the transition are given as follows:

**Game $\mathsf{G}_0$:** The vectors are computed according to the interaction:

$$\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}_{\mathbf{u}}^0}}_{\tilde{\mathcal{O}}_{\mathbf{v}}, \mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big)\ .$$

**Game $\mathsf{G}_1$:** We perform a computational basis change, making use of the randomness $r \xleftarrow{\$} \mathbb{Z}_q$ at coordinate $2N+1$ of $(\mathbf{u}_i)_{i=1}^H$ and of $\sigma_i \xleftarrow{\$} \mathbb{Z}_q$ at coordinate $2N+1$ of $(\mathbf{v}_i)_{i=1}^H$ so as to introduce a new non-zero $r' \xleftarrow{\$} \mathbb{Z}_q^*$ at coordinate $4N+4$ in $(\mathbf{u}_i)_{i=1}^H$ and secret sharings $(\tau_i)_{i=1}^H$ of 0 with only non-zero $\tau_i$ at coordinate $4N+4$ 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 $4N+4$ of $\mathbf{u}_i$, while keeping $\mathbf{v}_i^{(j)}[4N+4] = \mathbf{u}_{\ell,i}^{(\mathsf{rep})}[4N+4] = \mathbf{v}_{k,i}^{(\mathsf{rep})}[4N+4] = 0$. Given a $\mathsf{DSDH}$ instance $(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket c \rrbracket_1)$ in $\mathbb{G}_1$ where $\delta := c - ab$ is either 0 or 1, the basis changing matrices are:

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

All vectors changed under these bases are secret. We compute $\mathbf{B}_i$ using $\llbracket a \rrbracket_1$ and write the $\mathbf{u}$-vectors as follows:

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

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

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

If $\delta = 0$ we are in $\mathsf{G}_0$ else we are in $\mathsf{G}_{0.1}$, while updating $r$ to $r + br'$[10]. The difference in advantages is $|\Pr[\mathsf{G}_{0.1} = 1] - \Pr[\mathsf{G}_0 = 1]| \leq 2 \cdot \mathbf{Adv}_{\mathbb{G}_1}^{\mathsf{DDH}}(1^\lambda)$.

**Game $\mathsf{G}_{0.2}$:** We use $\mathsf{DSDH}$ 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 $\mathsf{DSDH}$ instance $(\llbracket a \rrbracket_2, \llbracket b \rrbracket_2, \llbracket c \rrbracket_2)$ in $\mathbb{G}_2$ where $\delta := c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

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

All vectors changed under these bases are secret. We compute $\mathbf{B}_i^*$ using $\llbracket a \rrbracket_2$ and write the $\mathbf{v}$-vectors as follows:

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

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:

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

[10] 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 swapping lemma for our $\mathsf{FH\text{-}DMCFE}$ scheme, this random $r$ is provided by the RO while hashing the tags.

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

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

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

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

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

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

**Game $\mathsf{G}_2$:** 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 $[1, N], [N+1, 2N]$ to $[2N+4, 3N+3], [3N+4, 4N+3]$ 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 $B_i$)

$$B_i = \begin{cases} B_i[row,col] & = 1 \text{ if } row = col\\ B_i[row,col] & = 1 \text{ if } (row,col) \in \{(2N+4+d, 1+d) : d \in [0, N-1]\}\\ B_i[row,col] & = 1 \text{ if } (row,col) \in \{(3N+4+d, N+1+d) : d \in [0, N-1]\}\\ B_i[row,col] & = 0 \text{ otherwise}\end{cases}$$

$$B'_i := \left(B_i^{-1}\right)^\top$$

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

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

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

We are in $\mathsf{G}_1$ in bases $(\mathbf{H}_i, \mathbf{H}^*_i)$ and in $\mathsf{G}_2$ 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 $\mathsf{G}_3$:** We perform a computational change to swap $\mathbf{x}_i$ from coordinate $[1, N]$ to $[2N+4, 3N+3]$ of $\mathbf{u}_i$, using the randomness $\rho_i$ at coordinate $2N+3$. We proceed by a sequence of $N+1$ hybrids, namely $\mathsf{G}_{2.k}$ for $k \in [0, N]$, such that $\mathsf{G}_{2.0} = \mathsf{G}_2$ and in $\mathsf{G}_{2.k}$ the first coordinates $[1, k]$ are swapped to $[2N+4, 2N+3+k]$, for $k \geq 1$. For $k \in [N]$, the transition from $\mathsf{G}_{2.k-1}$ to $\mathsf{G}_{2.k}$ is described below. Given a $\mathsf{DSDH}$ instance $(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket c \rrbracket_1)$ in $\mathbb{G}_1$ where $\delta := c - ab$ is either 0 or 1, the bases $(\mathbf{B}_i, \mathbf{B}_i^*)$ are changed following:

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

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

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

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

We cannot compute $\mathbf{b}_{i,1+k}^*$ and $\mathbf{b}_{1,2N+4+k-1}^*$ 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 $\mathbf{v}_i^{(j)}$, the same holds for $\mathbf{v}_{k,i}^{(\mathsf{rep})}$

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

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

The vectors, when we arrive at $\mathsf{G}_3$, are of the form:

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

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 $[2N+4, 3N+3]$ to coordinates $[3N+4, 4N+3]$ of $\mathbf{u}_i$, for all $i \in [H]$. The main idea is to consider the *selective* version $\mathsf{G}_j^*$ for $j \in \{4,5,6\}$, where the values $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N]}^{j\in[J]}$ 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

$$\Pr[\mathsf{G}_3^* = 1] = \Pr[\mathsf{G}_4^* = 1] = \Pr[\mathsf{G}_5^* = 1] = \Pr[\mathsf{G}_6^* = 1] \ . \tag{20}$$

In the end, we use a *complexity leveraging* argument to conclude that thanks to (20), 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 $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N]}^{j\in[J]}$, 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 $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N]}^{j\in[J]}$ and

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

where $\varDelta\mathbf{y}_i^{(j)} \coloneqq \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}$. Before elaborating the games, we note that for each $i \in [H]$, for all $j \in [J]$, the term

$$\langle \mathbf{x}_i, \varDelta\mathbf{y}_i^{(j)}\rangle = \sum_{k\in[N]} \mathbf{x}_i \varDelta\mathbf{y}_i^{(j)}[k] \tag{21}$$

is a constant. Otherwise there exists $\varnothing \neq I' \subseteq [H]$ and $j', j'' \in [J]$ so that

$$\sum_{i\in I'} \langle \mathbf{x}_i, \varDelta\mathbf{y}_i^{(j')}\rangle \neq \sum_{i\in I'} \langle \mathbf{x}_i, \varDelta\mathbf{y}_i^{(j'')}\rangle$$

while

$$\sum_{i\in[H]\setminus I'} \langle \mathbf{x}_i, \varDelta\mathbf{y}_i^{(j')}\rangle = \sum_{i\in[H]\setminus I'} \langle \mathbf{x}_i, \varDelta\mathbf{y}_i^{(j'')}\rangle \ ,$$

which contradicts 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 any $j \in [J]$. We describe the selective games below, starting from $\mathsf{G}_3^*$, where event $E$ is assumed true:

**Game** $\mathsf{G}_3^*$ : The selective version of $\mathsf{G}_3$, assuming event $E$ is true.

**Game** $\mathsf{G}_4^*$ : Knowing $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i\in[H],k\in[N]}^{j\in[J]}$ in advance, we perform a formal quotient on coordinates $[2N+4, 4N+3]$ of $\mathbf{u}_i$ and of $\mathbf{v}_i^{(j)}$. The bases are changed following:

$$B_i = \begin{cases} B_i[row, col] & = 1 \text{ if } row = col \leq 2N+3 \\ B_i[row, col] & = \frac{1}{\mathbf{x}_i[k]} \text{ if } \exists k \in [2N] : row = col = 2N+3+k \ \mathsf{AND}\ \mathbf{x}_i[k] \neq 0 \\ B_i[row, col] & = 1 \text{ if } \exists k \in [2N] : row = col = 2N+3+k \ \mathsf{AND}\ \mathbf{x}_i[k] = 0 \\ B_i[row, col] & = 1 \text{ if } row = col \geq 3N+3 \\ B_i[row, col] & = 0 \text{ otherwise} \end{cases} \tag{22}$$

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

We note that the matrices, which have dimensions $(4N+4)\times(4N+4)$, 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:

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

In summary, in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in $\mathsf{G}_3^*$, else we are in $\mathsf{G}_4^*$. The change is formal.

**Game** $\mathsf{G}_5^*$ : In this game we perform a formal basis change to move all the values 1 from coordinates $[2N+4, 3N+3]$ to coordinates $[3N+4, 4N+3]$ of $\mathbf{u}_i$. The bases are changed following:

$$B_i = \begin{cases} B_i[row, col] & = 1 \text{ if } row = col \\ B_i[row, col] & = \frac{1}{r'} \text{ if } \exists k \in [N] : (row, col) = (4N+4, 2N+3+k) \ \textsf{AND} \ \mathbf{x}_i[k] \neq 0 \\ B_i[row, col] & = 0 \text{ if } \exists k \in [N] : (row, col) = (4N+4, 2N+3+k) \ \textsf{AND} \ \mathbf{x}_i[k] = 0 \\ B_i[row, col] & = \frac{-1}{r'} \text{ if } \exists k \in [N] : (row, col) = (4N+4, 3N+3+k) \ \textsf{AND} \ \mathbf{x}_i[k] \neq 0 \\ B_i[row, col] & = 0 \text{ if } \exists k \in [N] : (row, col) = (4N+4, 3N+3+k) \ \textsf{AND} \ \mathbf{x}_i[k] = 0 \\ B_i[row, col] & = 0 \text{ otherwise} \end{cases} \tag{23}$$

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

We emphasize the the matrices are defined based on the knowledge of $(\mathbf{x}_i[k])_{i\in[H],k\in[N]}$ 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\text{-th left coord. 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]\mathbf{y}_i^{(1,j)}[N], \mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_i^{(0,j)}[N], \tau_i)_{\mathbf{H}_i^*} \\
&= (\cdots, \mathbf{x}_i[1]\mathbf{y}_i^{(1,j)}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_i^{(1,j)}[N], \mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_i^{(0,j)}[N], \\
&\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]\mathbf{y}_i^{(1,j)}[N], \mathbf{x}_i[1]\mathbf{y}_i^{(0,j)}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_i^{(0,j)}[N], \\
&\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]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N], \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N], 0)_{\mathbf{H}_i^*} \\
&= (\cdots, \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N], \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N], \\
&\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]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N], \mathbf{x}_i[1]\mathbf{y}_{k,i}^{(\mathsf{rep})}[1], \cdots, \mathbf{x}_i[N]\mathbf{y}_{k,i}^{(\mathsf{rep})}[N], 0)_{\mathbf{B}_i^*} \ .
\end{aligned}$$

In summary, in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in $\mathsf{G}_4^*$, else we are in $\mathsf{G}_5^*$. 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 observation (21) 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** $\mathsf{G}_6^*$ : We perform once again a formal quotient as from $\mathsf{G}_3^*$ to $\mathsf{G}_4^*$, on coordinates $[2N{+}4, 4N{+}3]$ of $\mathbf{u}_i$ and of $\mathbf{v}_i^{(j)}$. The bases are changed following:

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

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

We note that the matrices, which have dimensions $(4N+4) \times (4N+4)$, 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:

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

In summary, in bases $(\mathbf{H}_i, \mathbf{H}_i^*)$ we have the vectors as in $\mathsf{G}_5^*$, else we are in $\mathsf{G}_6^*$. The change is formal.

The above games demonstrate relation (20). 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 $\mathsf{ppt}$ adversary $\mathcal{A}$ in game $\mathsf{G}_t$. We build a $\mathsf{ppt}$ adversary $\mathcal{B}^*$ playing against $\mathsf{G}_t^*$ 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 $\gamma$.

The adversary $\mathcal{B}^*$ first guesses for the values $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i \in [H], k \in [N]}^{j \in [J]}$, 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 $(\mathbf{x}_i[k], \mathbf{y}_i^{(1,j)}, \mathbf{y}_i^{(0,j)})_{i \in [H], k \in [N]}^{j \in [J]}$ 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 $\mathsf{G}_t$ is perfect. Otherwise, $\mathcal{B}^*$ aborts the simulation and outputs a random bit $b'$. Since $E$

happens with some fixed probability $\gamma$ and is independent of the view of $\mathcal{A}$, we have:[11]

$$\begin{aligned}
\mathbf{Adv}^*_t(\mathcal{B}^*) &= \left|\Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1] - \frac{1}{2}\right| \\
&= \left|\Pr[E] \cdot \Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1 \mid E] + \frac{\Pr[\neg E]}{2} - \frac{1}{2}\right| \\
&= \left|\gamma \cdot \Pr[\mathsf{G}^*_t(\mathcal{B}^*) = 1 \mid E] + \frac{1-\gamma-1}{2}\right| \\
&\overset{(*)}{=} \gamma \cdot \left|\Pr[\mathsf{G}_t(\mathcal{A}) = 1] - \frac{1}{2}\right| = \gamma \cdot \mathbf{Adv}_t(\mathcal{A})
\end{aligned} \tag{24}$$

where $(*)$ comes from the fact that conditioned on $E$, $\mathcal{B}$ simulates perfectly $\mathsf{G}_t$ 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 $\mathsf{G}_6$, the vectors are now of the form:

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

We redo the computational swap from $\mathsf{G}_2$ to $\mathsf{G}_3$ so as to move $\mathbf{x}_i$ from coordinates $[3N+4, 4N+3]$ back to $[N+1, 2N]$ of $\mathbf{u}_i$. The calculation is similar, using a sequence of $N+1$ hybrids $\mathsf{G}_{6.k}$, namely $\mathsf{G}_{6.k}$ for $k \in [0, N]$, such that $\mathsf{G}_{6.0} = \mathsf{G}_6$ and in $\mathsf{G}_{6.k}$ the first coordinates $[3N+3, 3N+3+k]$ are swapped to $[N+1, N+k]$, for $k \geq 1$. For $k \in [N]$, the transition from $\mathsf{G}_{6.k-1}$ to $\mathsf{G}_{6.k}$ is described below. Given a $\mathsf{DSDH}$ instance $(\llbracket a \rrbracket_1, \llbracket b \rrbracket_1, \llbracket c \rrbracket_1)$ 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:

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

All vectors changed under these bases are secret. We compute $\mathbf{B}_i$ using $\llbracket a \rrbracket_1$ and write the $\mathbf{u}$-vectors as follows:

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

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

[11] This calculation (24) to relate $\mathbf{Adv}^*_t(\mathcal{B}^*)$ to $\mathbf{Adv}_t(\mathcal{A})$ is the core of our complexity levaraging argument, being built upon the previous information-theoretic game transtions and the probability of event $E$.

We cannot compute $\mathbf{b}^*_{i,N+k}$ and $\mathbf{b}^*_{1,3N+4+k-1}$ 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 $\mathbf{v}^{(\mathsf{rep})}_{k,i}$

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

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

$$\mathcal{A}^{\tilde{\mathcal{O}}_\mathbf{u}, \boxed{\mathcal{O}^1_\mathbf{u}}}_{\tilde{\mathcal{O}}_\mathbf{v}, \mathcal{O}_\mathbf{v}}\Big(1^\lambda, N, H, K, L, (J_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big)\ ,$$

implying $|\Pr[\mathsf{G}_8] - \Pr[\mathsf{G}_0]| \le (2N+8) \cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$ and the proof is completed. □

### C.6 Swapping with Repetitions – Proof of Lemma 34 (General Case)

**Lemma 34 (Swapping).** *Let $\lambda \in \mathbb{N}$ and $H = H(\lambda), K = K(\lambda), L = L(\lambda), J_i = J_i(\lambda), \widetilde{J}_i = \widetilde{J}_i(\lambda), N = N(\lambda) \in \mathbb{N}$ where $i \in [H]$ and $H, K, L, J_i, \widetilde{J}_i, N : \mathbb{N} \to \mathbb{N}$ are polynomials. Let $(\mathbf{B}_i, \mathbf{B}^*_i)$, for each $i \in [H]$, be a pair of random dual bases of dimension $4N+4$ in $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{t}, g_1, g_2, g_\mathsf{t}, \mathbf{e}, q)$. All basis vectors are kept secret. Let $R, R_1, \dots, R_K \in \mathbb{Z}_q$ be some public scalars. For $i \in [H]$, $\ell \in [L]$ and $k \in [K]$, sample $\sigma_i, \sigma_{i,k}, r, r_\ell \xleftarrow{\$} \mathbb{Z}_q$ conditioned on $\sum_{i\in[H]} \sigma_i = R$ and $\sum_{i\in[H]} \sigma_{k,i} = R_k$.*
*We consider the following oracles:*

$\underline{\tilde{\mathcal{O}}_\mathbf{u}}$**:** *On input $(\ell, i, \mathbf{x}^{(\mathsf{rep})}_{\ell,i}, \mathbf{x}'^{(\mathsf{rep})}_{\ell,i}) \in [L] \times [H] \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$, where $\mathsf{rep} \in [J_i]$ is a counter for the number of queries of the form $(\ell, i, \star, \star)$, sample $\rho^{(\mathsf{rep})}_{\ell,i} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{u}^{(\mathsf{rep})}_{\ell,i} = (\mathbf{x}^{(\mathsf{rep})}_{\ell,i},\ \mathbf{x}'^{(\mathsf{rep})}_{\ell,i},\ r_\ell,\ 0,\ \rho^{(\mathsf{rep})}_{\ell,i},\ 0^{2N+1})_{\mathbf{B}_i}\ .$$

$\underline{\boxed{\mathcal{O}^b_\mathbf{u}}}$**:** *For $b \in \{0,1\}$, on input $(i, \mathbf{x}_i^{(\tilde{j}_i)}) \in [H] \times \mathbb{Z}_q^N$, where $\tilde{j}_i \in [\tilde{J}_i]$ is a counter for the number of queries of the form $(i, \star)$, sample $\rho_i^{(\tilde{j}_i)} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\begin{aligned}&\textit{If } \boxed{b=0}: \quad \mathbf{u}_i^{(\tilde{j}_i)} = (\boxed{\mathbf{x}_i^{(\tilde{j}_i)}},\ \boxed{0^N},\ r,\ 0,\ \rho_i^{(\tilde{j}_i)},\ 0^{2N+1})_{\mathbf{B}_i}\\ &\textit{If } \boxed{b=1}: \quad \mathbf{u}_i^{(\tilde{j}_i)} = (\boxed{0^N},\ \boxed{\mathbf{x}_i^{(\tilde{j}_i)}},\ r,\ 0,\ \rho_i^{(\tilde{j}_i)},\ 0^{2N+1})_{\mathbf{B}_i}\ .\end{aligned}$$

$\underline{\mathcal{O}_\mathbf{v}}$**:** *On input $(i, \mathbf{y}_i^{(1,j_i)}, \mathbf{y}_i^{(0,j_i)}) \in [H] \times \mathbb{Z}_q^N \times \mathbb{Z}_q^N$, where $j_i \in [J_i]$ is a counter for the number of queries of the form $(i, \star, \star)$, sample $\pi_i^{(j_i)} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{v}_i^{(j)} = (\mathbf{y}_i^{(1,j_i)},\ \mathbf{y}_i^{(0,j_i)},\ \sigma_i,\ \pi_i^{(j_i)},\ 0,\ 0^{2N+1})_{\mathbf{B}^*_i}\ .$$

$\underline{\tilde{\mathcal{O}}_\mathbf{v}}$**:** *On inputs $(k, i, \mathbf{y}^{(\mathsf{rep})}_{k,i}) \in [K] \times [H] \times \mathbb{Z}_q$, where $\mathsf{rep} \in [J_i]$ is a counter for the number of queries of the form $(k, i, \star)$, sample $\pi^{(\mathsf{rep})}_{k,i} \xleftarrow{\$} \mathbb{Z}_q$ and output*

$$\mathbf{v}^{(\mathsf{rep})}_{k,i} = (\mathbf{y}^{(\mathsf{rep})}_{k,i},\ \mathbf{y}^{(\mathsf{rep})}_{k,i},\ \sigma_{k,i},\ \pi^{(\mathsf{rep})}_{k,i},\ 0,\ 0^{2N+1})_{\mathbf{B}^*_i}\ .$$

*If* $\sum_{i=1}^{H}\langle \mathbf{x}_i^{(\tilde{j}_i)}, \mathbf{y}_i^{(0,j_i)}\rangle = \sum_{i=1}^{H}\langle \mathbf{x}_i^{(\tilde{j}_i)}, \mathbf{y}_i^{(1,j_i)}\rangle$ *for all* $\tilde{j}_i \in [\widetilde{J}_i], j_i \in [J_i]$, *then the following advantage is negligible under the* SXDH *assumption:*

$$\Big| \Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}},\boxed{\mathcal{O}^0_{\mathbf{u}}}}_{\tilde{\mathcal{O}}_{\mathbf{v}},\mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \widetilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1]$$
$$- \Pr[\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}},\boxed{\mathcal{O}^1_{\mathbf{u}}}}_{\tilde{\mathcal{O}}_{\mathbf{v}},\mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \widetilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \to 1]\Big|$$
$$\leq (2N+8)\cdot \tilde{J}\cdot \mathbf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathbb{G}_2}(1^\lambda)$$

*where* $\tilde{J} = \max_{i\in[H]} \tilde{J}_i$ *and* $\mathcal{A}$ *can query the oracles* $\tilde{\mathcal{O}}_{\mathbf{u}},\boxed{\mathcal{O}^b_{\mathbf{u}}},\mathcal{O}_{\mathbf{v}},\tilde{\mathcal{O}}_{\mathbf{v}}$ adaptively, i.e. *the queries can be made in any order and any number of times respecting the (polynomial) upper bounds* $K, L, (J_i, \widetilde{J}_i)_{i\in[H]}$.

*Proof (Of Lemma 34).* We describe the games to change the vectors'generation as follows. For each $i \in [H]$, the value $J_i$ denotes the maximum number of possible repetitions $(\mathbf{u}_{\ell,i}^{(\mathsf{rep})})_{\mathsf{rep}}$, $(\mathbf{v}_i^{(j)})_j$, and $(\mathbf{v}_{k,i}^{(\mathsf{rep})})_{\mathsf{rep}}$, indexed by $\mathsf{rep}$ and $j$ over all $\ell, k$. We define $J \coloneqq \max i \in [H] J_i$.

**Game $\mathsf{G}_0$:** The vectors are computed according to the interaction:

$$\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}},\boxed{\mathcal{O}^0_{\mathbf{u}}}}_{\tilde{\mathcal{O}}_{\mathbf{v}},\mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \widetilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \ .$$

**Game $\mathsf{G}_1$:** We perform a sequence of hyrbids $\mathsf{G}_{0.\tilde{j}}$ for $\tilde{j} \in [0, \tilde{J}]$ where $\mathsf{G}_{0.0} = \mathsf{G}_0$ and in the $\tilde{j}$-th hybrid $\mathsf{G}_{0.\tilde{j}}$ we switch the first $\tilde{j}$ repetitions of $\mathbf{u}_i^{(\tilde{j})}$ (if $\tilde{J}_i < \tilde{j}$ there is no change on $\mathbf{u}_i$):

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

This means $\mathsf{G}_{0.J}$ has all vectors sampled from $D_1$ and the proof is completed. For $\tilde{j} \in [\tilde{J}]$, to go from $\mathsf{G}_{0.\tilde{j}-1}$ to $\mathsf{G}_{0.\tilde{j}}$ we apply Lemma 35. 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 35 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 35, noting that it is completely computational and we can modify only the $\tilde{j}$-th repetition:

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

where $r' \xleftarrow{\$} \mathbb{Z}_q^*$ and for all $j \in [J]$, the secret sharing $(\tilde{\tau}_i)_i$ in $\mathbf{v}_i$-vectors satisfies: $\sum_{i=1}^{H} \tilde{\tau}_i = 0$. We emphasize that the same share $\tilde{\tau}_i$ is used across all repetitions $j_i$ for a given $i$.

- $\underline{\mathsf{G}_{0.\tilde{j}-1.2}}$: We perform a formal duplication on *all* $\mathbf{v}$-vectors:

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

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 $[1, N]$ and $[3N + 4, 4N + 3]$ in $\mathbf{u}_i^{(\tilde{j})}$ using $(2N + 3)$-randomness. It is important that the change is computational using $\mathsf{DDH}$ 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 35.

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

- $\underline{\mathsf{G}_{0.\tilde{j}-1.4}}$: We now perform the complexity leveraging argument on coordinates $[2N+4, 4N+4]$, 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 35. 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 $(4N + 4)$-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 $[2N+4, 4N+4]$ with the *same* factor $(\mathbf{x}_i^{(\tilde{j})}[m])_m$. Later on, the formal swapping will modify the secret sharings $\tau_i$ 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 for any fixed $\tilde{j}, j$ thanks to condition
$$\sum_{i=1}^{H}\langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(0,j)}\rangle = \sum_{i=1}^{H}\langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)}\rangle$$
and moreover the updated $\tau_i$ does not depend on $\tilde{j}, j$ thanks to the fact that $\langle \mathbf{x}_i^{(\tilde{j})}, \mathbf{y}_i^{(1,j)} - \mathbf{y}_i^{(0,j)}\rangle$ is constant for all $\tilde{j}, j$, for any fixed $i \in [H]$. The argument can be made similarly as in the observation (21).

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

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

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 $\mathsf{G}_{0.\tilde{J}-1.4} \to \mathsf{G}_{0.\tilde{J}-1.5} = \mathsf{G}_{0.\tilde{J}}$ to save redundant security loss, while changing the definition of the hybrids.

After arriving at $\mathsf{G}_{0.\tilde{J}} = \mathsf{G}_1$, the vectors are computed following the interaction

$$\mathcal{A}^{\tilde{\mathcal{O}}_{\mathbf{u}}, \boxed{\mathcal{O}^1_{\mathbf{u}}}}_{\tilde{\mathcal{O}}_{\mathbf{v}}, \mathcal{O}_{\mathbf{v}}}\Big(1^\lambda, N, H, K, L, (J_i, \widetilde{J}_i)_{i\in[H]}, R, (R_k)_{k\in[K]}\Big) \ ,$$

the transitions are indistinguishable under $\mathsf{SXDH}$, and the proof is finished. □