Traitor Tracing in Multi-sender Setting
(TMCFE: Traceable Multi-client Functional Encryption)

Xuan Thanh Do[1], Dang Truong Mac[2], Ky Nguyen[3], Duong Hieu Phan[4], and Quoc-Huy Vu[5]

[1] Institute of Cryptography Science and Technology, Vietnam
[2] Vietnam National University, Hanoi, Vietnam
[3] DIENS, École normale supérieure, CNRS, Inria, PSL University, Paris, France
[4] LTCI, Telecom Paris, Institut Polytechnique de Paris, France
[5] De Vinci Higher Education, De Vinci Research Center, Paris, France

Abstract. Traitor tracing is a traditional cryptographic primitive designed for scenarios with multiple legitimate receivers. When the plaintext - that is, the output of decryption - is leaked and more than one legitimate receiver exists, it becomes imperative to identify the source of the leakage, a need that has motivated the development of traitor tracing techniques. Recent advances in standard encryption have enabled decryption outcomes to be defined in a fine-grained manner through the introduction of Functional Encryption (FE). Constructing FE schemes is intriguing, and achieving the tracing property adds an additional layer of complexity. Traitor tracing techniques have been actively developed for more than three decades, yet they have always remained within the same framework - a single sender responsible for encrypting all the data.

However, fine-grained decryption is particularly useful when data originates from multiple sources, allowing for joint computation on personal data. This leads to the concept of multi-client functional encryption (MCFE), where multiple concurrent senders independently encrypt their data while agreeing on the decryption of a specific function (e.g., a statistical measure) computed on the aggregated data, without revealing any additional information. In the era of cloud computing and big data, privacy-preserving joint computation is crucial, and tracing the source of any breach by dishonest participants becomes essential. Thus, in this paper we take the first step toward addressing the tracing problem in the general context of joint computation with multiple senders. Our contributions are twofold:

- **Conceptually:** We propose the first tracing model in the context of multi-sender encryption, namely *Traceable Multi-Client Functional Encryption* (TMCFE), which allows a pirate to extract secret information from both receivers and senders. Our model supports strong and naturally admissible decoders, removing artificial restrictions on the pirate decoder and thus addressing the shortcomings of existing traceable functional encryption schemes designed for the single-sender setting.
- **Technically:** To achieve our conceptual objective, we build upon the recently introduced notion of strong admissibility for MCFE. Our main technical contribution is a generic compiler that transforms a large class of MCFE schemes with weak admissibility into schemes with strong admissibility. This compiler not only helps overcome existing challenges but may also be of general interest within the functional encryption domain. Finally, we present a concrete lattice-based scheme TMCFE for inner-product functionalities that achieves post-quantum security under standard assumptions.

Table of Contents

1 Introduction

Traitor Tracing. Traitor tracing systems [CFN94] are a cryptographic technique designed to identify malicious users who leak secret decryption keys in a broadcast environment. It is particularly relevant in scenarios such as digital content protection, where content providers distribute encrypted media to legitimate subscribers but need to prevent unauthorized redistribution. If any collection of users (which are called as *receivers* in the context of network communication) attempts to create and sell a decoding box that can be used to decrypt the content, the tracing algorithm, given access to such pirate decoder, is guaranteed to identify at least one corrupt users, i.e., a member of those who contributed to the creation of the decoder.

In general, there are two main approaches to constructing traitor tracing schemes: the algebraic approach (e.g., group-based or lattice-based schemes) and the combinatorial approach (e.g., tree-based or collusion-secure code-based schemes). Algebraic schemes handle full collusion among traitors more effectively (e.g., [BSW06, BW06, BZ14, GKW18, GKW19, KW20, Zha20, AKYY23, GLW23, BLM+24]) or provide better efficiency in bounded-collusion settings (e.g., [BF99, LPSS14, ABP+17], among others). Meanwhile, combinatorial-approach schemes often provide better black-box tracing capabilities (e.g., [CFN94, BS95, BP08, BN08, DPY20, BPR24], among others). Many techniques were also developed to achieve systems with better efficiency [BZ14] and better security properties such as anonymity [NWZ16, GKW19] and transparency [BLM+24]. In the past decade, an important research direction has extended traitor tracing to more advanced encryption paradigms. These include group encryption [LYJP14], attribute-based encryption [CVW+18], and, more recently, functional encryption [DPP20, LAKWH22]. Throughout more than three decades of development, traitor tracing techniques, however, have remained within the one-to-many setting, where a single sender is responsible for encrypting all the data.

Fine-grained decryption. Recent advances in modern cryptography have enabled decryption outcome to be defined in a fine-grained manner, notably through the advent of functional encryption [BSW11]. Functional Encryption (FE) is an advanced encryption framework that enhances traditional public-key encryption by providing fine-grained control over the information that is revealed. Unlike traditional encryption schemes, which offer only an "all-or-nothing" decryption capability, where a legitimate decryption key of a user can extract the entire plaintext from a ciphertext while others learn nothing. On the other hand, FE allows an authorized user to obtain specific computed results from encrypted data without revealing any additional information. However, the initial form of functional encryption requires the entire dataset to be encrypted simultaneously by a single user or sender (i.e., following the one-to-many paradigm). For example, in the case of the inner-product functionality, all vectors must be encrypted at the same time by the same user using a common randomness. This constraint significantly limited real-world applicability—such as in aggregation scenarios—where data often comes from different sources and is collected over time. To address this limitation, multi-input and multi-client functional encryption schemes were proposed by [GGG+14], enabling independent encryption by multiple senders. Such multi-client functional encryption (MCFE) schemes, which allow multiple clients (i.e., senders) to encrypt data independently, can be considered as a (generalized) encryption scheme with fine-grained decryption in the arguably more complicated framework of multi-sender, i.e., the many-to-many communication.

On Private Key Encryption with Multi Clients Functional Encryption. Systematic studies on the context of multiple senders for functional encryption are initiated by the seminal works of [GGG+14, GKL+13, CDG+18a] that introduced and formalized the so-called notion of *multi-client functional encryption* (MCFE). In this setting, multiple clients act as senders, each encrypting *independently* their own data to obtain partial ciphertexts that can be jointly decrypted using a functional decryption key given by a trusted authority, when all partial ciphertexts share a same *tag*. This setting is particularly relevant in a growing number of applications where the encrypted data is collected from multiple independent sources[6], where there

[6] When there is a single sender knowing the function in advance, all-or-nothing public key encryption (PKE) is sufficient as one can encrypt directly the function evaluations of the data. However, in the multi-sender setting, even if there is *only one function* to be evaluated, the fact that multiple independent senders contibuting to the inputs

are scenarios that require primitives richer than usual PKE. Furthermore, the *private-key* version of MCFE where each client has a *secret encrption key*, is more relevant than its public-key counterpart, as argued in [GGG+14, CDG+18a]. This comes from an essential observation that: in case the encryption is public, an encryption of any target sender can be combined with encryptions of arbitrarily chosen values in place of other senders, and a functional decryption key, as long as the same tag is used for all ciphertexts. The amount of information leaked on the secret message of the target sender will be then too much, as different combinations can be done using public encryptions then decrypting with the functional decryption key. More specifically, this inherent leakage is captured by semantic security as follows. We say that an attack is *trivial* if correctness alone allows breaking the indistinguishability of the ciphertexts on $(x_i^{(0)})_i$ versus on $(x_i^{(1)})_i$ for all clients i, i.e., when the functional key on some function f gives $f(x_1^{(0)}, \ldots, x_n^{(0)}) \neq f(x_1^{(1)}, \ldots, x_n^{(1)})$. In case of public key encryption for MCFE, due to the above mix-and-match combination, to exclude trivial attacks, all functions f for which the functional key is known by the adversary must satisfy: $f(\ldots, x_i^{(0)}, \ldots) = f(\ldots, x_i^{(1)}, \ldots)$ for each encryption slot i. This is a very strong condition that excludes almost all non-trivial functions of a given class, and as a result the security notion becomes very weak. The private-key version of MCFE resolves this issue, and still makes sense in practical scenarios where the encryption key is a private information to each client.

Growing interests in MCFE is attested via a long line of works, in particular on realizing concrete function classes of inner products [CDG+18b, ABKW19, ABG19, LT19, CDSG+20, SV23, NPS24]. Along the way, fine-grained control of functional keys is also studied in [ACGU20, ATY23, Ngu24, NPP25, NPS25], together with further enhancements of the security notion [NPP23]. This shows a rich body of works on this setting of many-to-many private-key encryption of MCFE, where many senders (the clients) independently encrypt their data to be jointly decrypted by potentially many users in possession of functional decryption keys. The works [ATY23, Ngu24, NPP25, NPS25] ask and examine the question of how to control the functional keys in the private-key MCFE setting where only authorized users can decrypt, leaving thus space for further investigation on other important aspects, including tracing colluded decryption keys.

In this work, we envision a new model for traitor tracing in the multi-sender setting, and we propose the first traitor tracing model for MCFE, that we refer to as *traceable MCFE*. We remark that the problem of traitor tracing for many-to-many communication is interesting in its own right. Traitor tracing then comes up naturally in virtually any application where multi-client functional encryption is applicable, especially in the era of cloud computing and big data, where privacy-preserving joint computation is crucial. Unlike the one-to-many setting, MCFE introduces additional challenges due to the presence of multiple parties with distinct roles (e.g., senders, receivers, or both). Collusion between adversarial entities occupying different roles could facilitate the construction of stronger pirate decoders, potentially rendering tracing infeasible. This observation is critical within the context of MCFE, where encryption remains private, and the compromise of secret encryption keys introduces further security risks.

Consequently, simply adapting existing techniques from the one-to-many traitor tracing paradigm is insufficient for designing an MCFE scheme that incorporates tracing capabilities. Our work aims to address these challenges by establishing a rigorous framework for traitor tracing in MCFE and proposing a construction that ensures tracing mechanism in the many-to-many communication setting.

1.1 Our Contributions

We construct traitor tracing systems for multi-client functional encryption for inner-product functionality from standard lattice assumptions, achieving anonymity of honest users and bounded collusion. Our main contributions are summarized as follows.

(1) A new model for traitor tracing in the multi-sender setting. We introduce the notion of traceable multi-client functional encryption (TMCFE) as a new model to build traitor tracing systems in the many-to-many communication setting. In particular, we propose a standard and natural security model for TMCFE.

of the function means one cannot naively use PKE because the function evaluation is not known at independent encryption time among senders.

A natural starting point for defining such a system is the notion of traceable functional encryption, which was first introduced in [DPP20]. [DPP20] also gives a concrete, efficient construction for the inner-product functionality, however, their construction only achieves a very weak security of one-target security for tracing. By adapting the technique of Agrawal et al. [ABP+17] into the functional encryption setting, a subsequent work of [LAKWH22] proposes traceable functional encryption for inner-product schemes that offer stronger security, specifically, multi-target security (referred to as adaptive tracing security). However, their security notion still suffer from a major limitations: it introduces certain artificial conditions or restrictions that do not align well with practical scenarios—making it incapable of tracing even relatively simple pirate decoders. We discuss more on these limitations in Section 1.2. Our new security model for tracing not only improves previous tracing security from previous works and but also extend it to the multi-sender setting.

Along the way, we also provide characterizations of our definition for TMCFE, and we show that public tracing is impossible in general for MCFE. Interestingly, our impossibility exploits the possibility of corruption of both senders (with private-key encryption) and receivers in the context of MCFE. Informally, our impossibility says that public tracing and IND-CPA security cannot be achieve simultaneously.

(2) Bounded-Collusion Embedded Identities TMCFE for Inner-Product. We present a construction of TMCFE for inner-product with bounded-collusion and embedded identities, based on standard Learning-With-Errors assumptions. Informally, an embedded identities traitor tracing scheme preserves honest receivers' privacy from the senders. While there are several traitor tracing schemes for PKE with bounded-collusion and embedded identities that we can leverage, adapting these constructions to the MCFE setting requires new cryptographic techniques. Furthermore, since (MC)FE for general circuits implies the existence of indistinguishability obfuscation, which is far from being practical and we are interested in constructions that can be based on standard assumptions with concrete efficiency, in this work, we focus on the construction of less general functionalities which are still expressive enough for practical scenarios, namely, the inner-product functionality.

(3) Enhancing tracing security for TMCFE from strong IND-CPA security. We identify a close connection between the admissibility condition required[7] in the IND-CPA security definition for MCFE and the tracing security definition of TMCFE. Traceability and security have often been considered separately in the context of traitor tracing for public key encryption (PKE). However, our work indicate that improving one might enhance the other, and that the two are closely related in the context of TMCFE.

MCFE schemes for inner-product functionality typically adopt *weak* admissibility conditions due to the deterministic nature of the encryption scheme [CDG+18a, LT19, ABG19], whereas an optimal and stronger admissibility conditions for MCFE are only recently studied in the work of [NPP23]. Importantly, this *strong* admissibility conditions of [NPP23] leads to a stronger IND-CPA security. On the other hand, we also show that the same *strong* admissibility conditions are necessary for a robust tracing model, allowing the removal of artificial restrictions present in prior works, such as those discussed in [DPP20] and [LAKWH22]. To that end, we also construct a generic compiler that transforms MCFE for inner-product under certain structure of decryption[8] from weak admissibility to strong admissibility, with little overhead.

We expect the same relationship would hold in a broader context of multi-sender with fine-grained decryption (for example, decentralized MCFE [CDG+18a]), and leave these interesting questions as open for future work.

1.2 Technical Overview

The standard IND-CPA security model for MCFE [GGG+14, GKL+13, CDG+18a] consists of a security experiment between a challenger and an adversary, where the adversary is given access to a number of oracles **(Initialise, Extract, Enc, Corrupt)** and a challenge oracle **LoR** to which they can make queries. The names of oracle are self-explanatory, in particular, the adversary can query to **Corrupt** to obtain the secret

[7] Admissibility conditions are required to prevent trivial attacks in IND-CPA for MCFE.
[8] All known constructions of MCFE from lattice assumptions satisfy these requirements.

encryption keys, *separately* from the corruption of the decryption keys that are associated with functions, which can be obtained via **Extract**. We remark that as MCFE/MIFE are secret-key primitives, there is an **Enc** oracle to which the adversary can ask for any encryption of its choice for each encryption slot. The ability to corrupt makes the adversary stronger and thus leads to a stronger notion of security, as reflected via the notion of *admissibility* conditions for adversary. The admissibility conditions will be discussed later in this sequel when it is relevant.

1.2.1 Definition of traitor tracing in the multi-client setting

We first motivate a particular type of traitor tracing with *embedded identities*. Embedded identities, first studied by Nishimaki, Wichs, and Zhandry [NWZ16], enable the inclusion of arbitrary information within secret keys. Ideally, a tracer would want to identify the traitor in order to prosecute or fine. Without embedded identities (i.e., in an indexed traitor tracing system), the key issuer could maintain an explicit mapping (as a look-up table) between the user identification information and the indices of their respective decryption keys. Embedded identities address this issue by directly incorporating the necessary information into the issued keys, eliminating the need for such a mapping. Our goal is then to construct embedded identity traitor tracing (EITT) for multi-client functional encryption for the class of inner products (IPMCFE).

Traceable Multi-Client Functional Encryption. In the following, we refer to an encryptor as a *"client"* and a decryptor as a *"user"*. We recall that our setting is *multi-client*, for independent encryption of partial ciphertexts, with *multi-receiver* where each receiver can request for functional decryption keys that allow decrypting clients' ciphertexts under the same tag. The tracing adversary is allowed to obtain the secret encryption keys by corrupting the clients, *separately* from the corruption of the decryption keys that are associated with functions.

The first tracing model: Single-sender, One-target [DPP20]. The concept of *traceable* functional encryption (FE) was first explored in [DPP20], where the authors introduced traceable inner product functional encryption (TIPFE). In their construction, decryption keys are associated with tuples $(j, \mathbf{y})$ representing user indices and functional vectors. The ciphertexts are computed for some vector $\mathbf{x}$ in such a manner that the decryption reveals nothing about the message $\mathbf{x}$ except the inner product $\langle \mathbf{x}, \mathbf{y} \rangle$. However, they only achieve weak security of one-target black-box traceability, which imposes significant restrictions on the adversary: (1) the adversary must commit to a single target function $\mathbf{y}^*$ before it sees the public key; (2) it is limited to requesting decryption keys solely for $\mathbf{y}^*$; and (3) it outputs a black-box distinguisher $D_{\mathbf{y}^*}$ associated with $\mathbf{y}^*$ and two messages $(\mathbf{x}^{(0)} \neq \mathbf{x}^{(1)})$ such that $\langle \mathbf{x}^{(0)}, \mathbf{y} \rangle \neq \langle \mathbf{x}^{(1)}, \mathbf{y} \rangle$. Throughout this sequel, we call a functional vector $\mathbf{y}$ as *differentiating* corresponding to a pair of messages $(\mathbf{x}^{(0)} \neq \mathbf{x}^{(1)})$ if $\langle \mathbf{x}^{(0)}, \mathbf{y} \rangle \neq \langle \mathbf{x}^{(1)}, \mathbf{y} \rangle$ and *non-differentiating* otherwise. It is worth noting that it was argued in [DPP20] that one-target traitor tracing for FE is already stronger than (indexed) bounded traitor tracing for plain PKE.

The second tracing model: Restrained Adversary, Single-sender, Multi-target [LAKWH22]. A subsequent improvement by [LAKWH22] introduced many-target black-box traceability (also referred to as adaptive traceability) for TIPFE. This enhanced security model allows the adversary to request multiple decryption keys adaptively for a set of functional vectors $\{\mathbf{y}_i\}_i$, while still requiring that each $\mathbf{y}_i$ is differentiating with respect to the chosen message pair. Specifically, the adversary must output a black-box distinguisher D and two messages $(\mathbf{x}^{(0)} \neq \mathbf{x}^{(1)})$ such that all queried vectors $\mathbf{y}_i$ satisfy $\langle \mathbf{x}^{(0)}, \mathbf{y}_i \rangle \neq \langle \mathbf{x}^{(1)}, \mathbf{y}_i \rangle$ for all i. Despite this improvement from [LAKWH22] compared to [DPP20], we argue that the traceability definition in [LAKWH22] is overly restrictive and may fail to capture practical pirate decoders. Consider an adversary that produces a distinguisher D associated with a functional key for $\mathbf{y}$ as $\mathbf{y} = \mathbf{y}_1 + \mathbf{y}_2$, where $\mathbf{y}$ is created from a key for $\mathbf{y}_1$, which is *non-differentiating* on a tracing signal $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, *i.e.* $\langle \mathbf{x}^{(0)}, \mathbf{y}_1 \rangle = \langle \mathbf{x}^{(1)}, \mathbf{y}_1 \rangle$, together with a *differentiating* key for $\mathbf{y}_2$, *i.e.* $\langle \mathbf{x}^{(0)}, \mathbf{y}_2 \rangle \neq \langle \mathbf{x}^{(1)}, \mathbf{y}_2 \rangle$. Since the adversary asked for a non-differentiating key $\mathbf{y}_2$, it is deemed inadmissible in the tracing game under [LAKWH22]'s definition, even though the output distinguisher D remains effective in distinguishing $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ with high probability. This counterexample

suggests that a natural and stronger security notion should allow the adversary to obtain functional decryption keys without artificial constraints, thereby enabling a more robust traitor tracing for functional encryption.

OUR TRACING MODEL: MULTI-SENDER, MULTI-TARGET, NO RESTRICTION. We emphasize that the foregoing discussion applies to the works on *single-sender* FE [DPP20, LAKWH22, ZZ23] and we will extend this to the *multi-sender* setting, henceforth called *multi-client* following the standard of the FE community, in our work. This is demonstrated in our below definitional choices.

Building upon the above insights, we extend traitor-tracing functional encryption to the multi-client setting, with *embedded identities*. We propose a standard and natural model for traceable IPMCFE as follows. A traitor tracing system for IPMCFE with n clients consists of five poly-time algorithms – $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}$ and Trace, which expands with the new Trace into the standard syntax of MCFE. The Setup algorithm takes as input security parameter λ, the identity space $\mathcal{ID}$, and generates a master secret key msk, encryption keys $(\mathsf{ek}_i)_{i\in[n]}$ and a tracing key tk. The key generation algorithm KeyGen, on input an identity id and a functional vector $\mathbf{Y}$, where $\mathbf{Y} = [\mathbf{y}_1\| \cdots \|\mathbf{y}_n]$, outputs a decryption key $\mathsf{sk}_{\mathsf{id},\mathbf{Y}}$. Each client can use the encryption algorithm Enc to encrypt a vector message $\mathbf{x}_i$ using encryption key ek_i. For *correctness*, given a list of ciphertexts of all n clients under the same tag, the decryption algorithm can decrypt using any one of the decryption key $\mathsf{sk}_{\mathsf{id},\mathbf{Y}}$ to obtain $\langle \mathbf{X}, \mathbf{Y}\rangle$ where $\mathbf{X} = [\mathbf{x}_1\| \cdots \|\mathbf{x}_n]$. The tracing algorithm Trace takes tracing key tk, two adversarially chosen challenges $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, a challenge tag and some success parameter y, and is given (black-box) oracle access to a pirate decoding algorithm D:

$$\mathsf{Trace}^{D}(\mathsf{tk}, 1^{y}, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \ .$$

It outputs a set $S \subseteq \mathcal{ID}$ of users signaling that the keys sk_j for $j \in S$ were used to create the pirate decoder D. A crucial distinction between MCFE and public-key (functional) encryption lies in the private nature of encryption keys: each client retains its encryption key privately, and the adversary is not only allowed to request decryption keys (corrupting multiple users) but also allowed to corrupt the clients to obtain their encryption keys (corrupting multiple senders). Two essential *security properties* include the *security against chosen-plaintext attacks* (IND-CPA) and the *tracing security*. The IND-CPA follows as that of the "vanilla" MCFE. The *tracing security* of the system states that: *(i)* the tracing algorithm outputs *with high probability* at least one identity from the set of colluding users, and *(ii)* without false accusations of honest users, except with negligible probability. We elaborate the notable design choices in our modeling below.

Strong Admissibility for IND-CPA. The standard approach to prove tracing security is to reduce it to the IND-CPA security of the encryption scheme. At the heart of the IND-CPA is the notion of *admissibility* condition for adversary, as with all security models for advanced encryption notion, where we have to exclude attacks that trivially use the nature of the function class to break *any efficient schemes*. An attack is *admissible* if it does require the specific details of the scheme to succeed. The evolution of how *admissible* an adversary can be with respect to the security notion is highlighted in the seminal work on MCFE [CDG+18a] and follow-up works to enhance MCFE with a more fine-grained corruption model [AGT21, NPP23]. In particular, given the **Corrupt** oracle for the adversary to corrupt the secret encryption keys of some client(s) i,

1. As the first generation of MCFE, security definitions in [CDG+18a] and a lot of the follow-ups [CDG+18b, ABKW19, ABG19, LT19, CDSG+20, AGT21] imposed a common constraint on the adversary such that $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{(1)}$ whenever client i is corrupted.
2. Very recently, the above condition was revised in [NPP23], which introduced a stronger and provably optimal admissibility condition requiring that $\langle \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(1)}, \mathbf{y}_i\rangle = 0$ for any corrupted client i. This condition allows more attack to be considered, thus making the security model more robust[9]. This already launches a new generation of stronger MCFE schemes [NPP25].

[9] To see the difference, the *weak* admissibility condition (Item 1) allows *deterministic encryption*, which is not justified if we are to aim for IND-CPA, while the *strong* admissibility condition (Item 2) necessitates *probabilistic encryption*.

In this paper we refer to the admissibility condition in [NPP23] (Item 2) as *strong admissibility*, meanwhile the admissibility condition in [CDG+18a] (Item 1) as *weak admissibility*. It turns out that this strong (and optimal) admissibility notion studied in [NPP23] (Item 2) is essential for a meaningful notion of traitor tracing FE in the multi-client setting. Let us consider the extreme case of *one* client and where the adversary corrupts this client, thereby obtaining the secret encryption key ek. Suppose the MCFE is IND-CPA secure, we consider different cases *as per* the admissibility condition and want to perform a reduction from tracing security to IND-CPA security, given in particular pair of distinct vectors ($\mathbf{x}^{(0)} \neq \mathbf{x}^{(1)}$) for the *only slot* (there is one client) that the tracing adversary chooses:

- Under the weak admissibility condition (Item 1), either *(i)* our reduction does *not* corrupt ek and forward to the IND-CPA challenge oracle **LoR** and relies on the tracing adversary to break the resulted challenge ciphertext, or *(ii)* our reduction *corrupts* ek (for instance, when the tracing adversary wants to corrupt ek themselves) and is thus bound to the weak admissibility condition to submit only identical messages to the IND-CPA challenge oracle **LoR**. The first case *(i)* leads to a winning only with negligible probability as the tracing adversary is acting to break the IND-CPA security themselves, the second case *(ii)* ensues an IND-CPA advantage no better than naive guessing $\frac{1}{2}$ because the **LoR** returns a challenge ciphertext that will be independent of the challenge bit (as the messages are identical).
- Under the strong admissibility condition (Item 2), the reduction can be done straightforwardly as we can just forward the tracing adversary's challenge to the IND-CPA challenge oracle **LoR** and and send the resulted challenge ciphertext to the tracing adversary.

Moreover, from an application point of view, the corruption of encryption keys is a realistic scenario in the context of traitor tracing, as it models the fact that malicious clients, who in conjunction with colluding users, can play the role of creating pirate decoders (so that the decoder is *good* at distinguishing).

Black-Box but Private Tracing. Another aspect of the tracing security is the *black-box* nature of the tracing algorithm, that is, the tracing algorithm ignores the implementation of the pirate decoder, only makes queries to the decoder and observes the outputs. This limits the information that is available to the tracing algorithm, and thus makes the tracing security more resilient. Moreover, it turns out that this choice is also inherent to our current setting of *multi-client* functional encryption, where the encryption keys are private to the clients. More specifically, we show that, given function classes with some *solvability* properties (see Definition 14 for formal definition), public tracing with black-box access to the pirate decoder is not possible, assuming the encryption scheme is IND-CPA secure.

Theorem 1 (Informal). *There exist non-trivial function classes for joint computation of at least* two *clients, including the class of inner products, for all traitor tracing MCFE scheme for any of these classes, supporting at least* two *clients, if the scheme is* correct *and* traceable *with* public black-box *tracing, then it* cannot be IND-CPA secure *with strong admissibility.*

Two strengthening points of our theorem are worth noting: *(i)* - our IND-CPA adversary does *not* need to corrupt even the encryption keys in order to run the *public* tracer (otherwise the tracing algorithm cannot be *public* as it depends on secret information in this setting of *private* encryption keys for multi-clients), and *(ii)* - the tracing succeeds even when running with *black box* access to the decoder. Point *(i)* means our theorem extends even to more restrictive corruption settings where no encryption key can be corrupted by the tracing adversary (ours allows both separated corruption of encryption *and* decryption keys). Point *(ii)* only strengthens our theorem to cover impossibility in *white box*[10] public tracing, as all we rely on is the traced results from the *public* tracer. As a corollary of Theorem 1, we have to restrict the tracing algorithm to be *private* in order to have a secure traitor tracing MCFE scheme.

1.2.2 Strong Admissibility with Fine-Grained Access Control in MCFE

We argue in our discussions above that the necessary condition for our strong tracing model is an IND-CPA security with strong admissibility ([NPP23], Item 2). It turns out that our concrete traitor tracing MCFE

[10] The notion of white-box tracing is defined in [Zha21].

construction for the function class computing inner products will use, as a building block, a MCFE scheme for a richer class of functions and satisfying the strong admissibility condition. This function class of the underlying building block is *integrating access control into inner product computation* (see Definition 1), as initiated in previous works [ACGU20, NPP22, NPP25], *i.e.*, following the definitional framework of [NPP22, NPP25]: (†) - Each client encrypts their individual vector $\mathbf{x}_i$ together with some attributes $\mathsf{ac\text{-}ct}_i$. (‡) - The functional key is associated with $\mathbf{Y} = [\mathbf{y}_1 \| \cdots \| \mathbf{y}_n]$ and some policy $\mathsf{ac\text{-}k}$. ($\diamond$) - The decryption algorithm outputs $\langle \mathbf{X}, \mathbf{Y} \rangle$ if the policy $\mathsf{ac\text{-}k}$ is satisfied by the attributes $(\mathsf{ac\text{-}ct}_i)_{i=1}^n$, where $\mathbf{X} = [\mathbf{x}_1 \| \cdots \| \mathbf{x}_n]$ being the concatenation of the clients' vectors. Putting forth our requirements for the final traitor tracing MCFE, we need the access control to be as general as bounded depth circuits, *e.g.* as for the classical attribute-based constructions from [BGG+14]. Existing works on MCFE for this function class of *attribute-based inner products* are either not satisfying the strong admissibility condition [ACGU20, NPP22] or do not meet the expressiveness for the access control [NPP25]. In order to handle this challenge, we proceed by two steps:

Solution - Step 1. We generalize the lattice-based techniques from [LLW21] to handle access control with inner products [ALS16] for single-client. Then we generalize the transformation of [ABG19] to obtain a MCFE scheme that satisfies IND-CPA under the *weak* admissibility condition. This construction is presented in Section 8.

Solution - Step 2. As a complete novelty, we provide a new transformation that upgrades the MCFE scheme from **Step 1** to satisfy the *strong* admissibility condition (Item 2). Our transformation works for the general case of access control with inner products, preserving the asymptotic efficiency of the underlying MCFE with weak admissibility and turning it into a MCFE with strong admissibility.

One of our main technical contributions is our ideas for solving **Step 2**. For the rest of the technical overview, we thus essentially focus on **Step 2** below, after briefly explaining how to achieve **Step 1**.

Admissibility Conditions for Inner Product with Access Control. We first translate below an informal version of the *strong* admissibility for for inner product with access control, followed the work of [NPP23]. The formal conditions are given in Remark 1.

S.1 For all challenge vectors data-attribute $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all function-policy $((\mathbf{y}_i)_{i\in[n]}, \mathsf{ac\text{-}k})$ to **KeyGen**, for all corrupted i, up to repetitions, it holds that: $\langle \mathbf{x}_i^{(b)} - \mathbf{x}_i^{(1)}, \mathbf{y}_i \rangle = 0$.

S.2 For all challenge vectors data-attribute $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all function-policy $((\mathbf{y}_i)_{i\in[n]}, \mathsf{ac\text{-}k})$ to **KeyGen** that can decrypt the challenge ciphertexts, *i.e.* the policy $\mathsf{ac\text{-}k}$ is satisfied by the attributes $(\mathsf{ac\text{-}ct}_i)_{i\in[n]}$, summing over all *honest* i, up to repetitions of $\mathbf{x}_i$ at each i[11], does *not* differ between challenge bit $b = 0$ and $b = 1$.

When considering the *weak* admissibility condition from [CDG+18a], condition **S.1** is replaced by

W.1 For all challenge vectors data-attribute $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all *corrupted* i, the two challenge components at i, up to repetitions, are equal: $\mathbf{x}_i^{(b)} = \mathbf{x}_i^{(1)}$.

At first glance, the *strong* admissibility condition **S.1** is *less restrictive* than the *weak* admissibility condition **W.1**, thus leading to a more robust security notion that can cover *more attacks*.

In short, each client i in the MCFE scheme $\mathcal{E}^{\mathsf{w}}$, encrypts their $\mathsf{ek}_i, \mathbf{x}_i, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i)$ using the encryption $\mathsf{Enc}^{\mathsf{one\text{-}slot}}(\mathsf{pp}_i, \star)$ from [LLW21] by:

$$\begin{aligned}
&\mathsf{ct}_{i,\mathsf{tag}} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{Enc}(\mathsf{ek}_i = (\mathsf{pp}_i^{\mathsf{one\text{-}slot}}, \{\mathsf{k}_{i,j}\}_j), \mathbf{x}_i, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i)) = \mathcal{E}^{\mathsf{one\text{-}slot}}.\mathsf{Enc}(\mathsf{pp}_i^{\mathsf{one\text{-}slot}}, \mathbf{w}_i) \ , \\
&\quad \text{where} \quad \mathbf{w}_i := [\mathbf{0} \,\|\, \ldots \,\|\, \mathbf{0} \,\|\, \mathbf{x}_i \,\|\, \mathbf{0} \,\|\, \ldots \,\|\, \mathbf{0}] + \mathbf{t}_{i,\mathsf{tag}} \ , \\
&\text{with keys:} \quad \mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{KeyExtract}(\mathsf{msk} = \mathsf{msk}^{\mathsf{one\text{-}slot}}, \mathbf{Y}, \mathsf{ac\text{-}k}) = \mathcal{E}^{\mathsf{one\text{-}slot}}.\mathsf{KeyGen}(\mathsf{msk}, \mathbf{Y}, \mathsf{ac\text{-}k}) \ , \quad (1)
\end{aligned}$$

[11] This is a strong security guarantee that allows the adversary to reuse the tag at slots i, which is considered standard in the context of MCFE [AGT21, NPS25, NPP25].

and $\mathbf{t}_{i,\mathsf{tag}} \leftarrow \sum_{j\neq i}(-1)^{j<i}\mathsf{PRF}(\mathsf{k}_{i,j},\mathsf{tag})$. The ek_i is private for each i so that the PRF keys are secret. The *correctness* can be verified when ever the attributes $(\mathsf{ac\text{-}ct}_i)_i$ satisfy the policy $\mathsf{ac\text{-}k}$ in a key $((\mathbf{y}_i)_i, \mathsf{ac\text{-}k})$. The CPA-security is ensured by the CPA-security of the underlying single-client FE scheme for inner products with access control, after applying the PRF security to the masks. Last but not least, a sophisticated proof strategy can guarantee the *dynamic* corruption (of multi clients) security, which we also use ubiquitously in this work.

From Weak Admissibility (W.1) to Strong Admissibility (S.2). The reasons that $\mathcal{E}^{\mathsf{w}}$ above only achieves security under admissibility condition **W.1** are: *(I)* - when i is corrupted, the PRF masks are no longer pseudorandom and we lose the effectiveness of the secret sharings $t_{i,\mathsf{tag}}$ of 0. *(II)* - if one relaxes condition **W.1** aiming for the stronger security with **S.1**, an adversary can *(IIa)* - query $\mathbf{x}_i^{(0)} \neq \mathbf{x}_i^{(1)}$ to $\mathbf{LoR}$, *(IIb)* - obtaining the challenge $\mathsf{ct}_{i,\mathsf{tag}}$, *(IIc)* - makes use of *the partial decryptions of* $\mathcal{E}^{\mathsf{one\text{-}slot}}$ knowing the components of $\mathsf{ct}_{i,\mathsf{tag}}$ contains $\mathcal{E}^{\mathsf{one\text{-}slot}}.\mathsf{Enc}(\mathsf{pp}_i^{\mathsf{one\text{-}slot}}, \mathbf{w}_i) = \mathcal{E}^{\mathsf{one\text{-}slot}}.\mathsf{Enc}(\mathsf{pp}_i^{\mathsf{one\text{-}slot}}, [\mathbf{0} \parallel \ldots \parallel \mathbf{x}_i \parallel \ldots \parallel \mathbf{0}] + \mathbf{t}_{i,\mathsf{tag}})$, and *(IId)* - the preceding partial decryption leaks information about $\langle \mathbf{x}_i^{(b)}, \mathbf{y}_i\rangle$ after the adversary makes use of point *(I)*. Hence, any relaxation of condition **W.1** to **S.1** will allow the adversary to choose $\mathbf{x}_i^{(0)} \neq \mathbf{x}_i^{(1)}$ so that $\langle \mathbf{x}_i^{(0)}, \mathbf{y}_i\rangle \neq \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i\rangle$ to win the IND-CPA game.

Our Solution towards Admissibility (**S.1**,**S.2**): Randomisation by a Layer of Inner Product FE. We now describe the high-level idea of our transformation in **Step 2**, important ideas/observations are *underlined*. From the aforementioned overview of our **Step 1**, the resultant MCFE can only be secure under the *weak* admissibility condition, that is, under conditions **S.2** (summing over honest clients does not trivially break the scheme) and **W.1** (*as per* weak admissibility, on any corrupt i the adversary can only use identical challenges). Our transformation lifts condition **W.1** to the *strong* admissibility condition **S.1**. Our important observation is that the two structural properties: *encryption and decryption can be done partially, before a global finally linear combination* and *two-step decryption* (a formal definition can be found in [ABG19, Def. 3.1]) are *preserved* into the resultant MCFE, *cf.* the encryption in Equation (1). In particular, we will make use the *linearity that is preserved throughout*. From the above exposition of the challenge regarding $\mathcal{E}^{\mathsf{w}}$ from the attack *(II)*, our cornerstone idea is *to randomise the component* $\mathbf{w}_i$ in the encryption of $\mathcal{E}^{\mathsf{w}}$ not relying entirely on the PRF masks $\mathbf{t}_{i,\mathsf{tag}}$. In the following we describe our ideas at the level of $\mathcal{E}^{\mathsf{w}}$ and do not go lower to $\mathcal{E}^{\mathsf{one\text{-}slot}}$ (we recall that $\mathcal{E}^{\mathsf{one\text{-}slot}}$ is instantiated from [LLW21]). Roughly, we modify $\mathcal{E}^{\mathsf{w}}$ step-by-step as follows, new modifications are boxed:

- During $\mathcal{E}^{\mathsf{w}}.\mathsf{Enc}$ for client i, on $\mathbf{x}_i$ with $\mathsf{ac\text{-}ct}_i$, we first sample random r_i, add in-place a randomness to mask $\mathbf{x}_i$: $\mathbf{w}_i = [\mathbf{0} \parallel \ldots \parallel \mathbf{x}_i - \boxed{r_i \cdot \mathbf{v}_i} \parallel \ldots \parallel \mathbf{0}] + \mathbf{t}_{i,\mathsf{tag}}$ where $\boxed{\mathbf{v}_i}$ is another secret specific for each client i, generated at setup time.
- This quantity of random $r_i \cdot \mathbf{v}_i$ induces intuitively deviated values $\langle \mathbf{x}_i, \mathbf{y}_i\rangle - \boxed{r_i \cdot \langle \mathbf{v}_i, \mathbf{y}_i\rangle}$, given the function component $\mathbf{y}_i$ at client i, during the partial decryption *as per* the two-step decryption property. We come up with the idea of *using another layer of public key IPFE* (inner product FE) for each client i so as to cancel out, thanks to the linearity, this error $\boxed{r_i \cdot \langle \mathbf{v}_i, \mathbf{y}_i\rangle}$ during the process of two-step decryption.
- To employ inner product FE, we rewrite $\boxed{r_i \cdot \langle \mathbf{v}_i, \mathbf{y}_i\rangle} = \langle r_i \cdot \mathbf{t}_i, (\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}_i\rangle$, where N is the length of the each slot, and $\mathbf{t}_i$ contains non-zero values and is secret to each i sampled at setup time, where "$\circ$" is the Hadamard product. The part $r_i \cdot \mathbf{t}_i$ can be now dealt with at encryption time by a ciphertext of the IPFE scheme, the key for $\boxed{(\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}}$ is generated by the IPFE scheme, during the key generation for the MCFE scheme knowing $\mathbf{y}_i$ and the master secret key containing $\mathbf{v}_i, \mathbf{t}_i$. Using some IPFE $\mathcal{E}^{\mathsf{als\text{-}ip}}$, the new encryption of the resultant $\mathcal{E}$ at i now looks like: r_i is random

$$\mathsf{ct}_{i,\mathsf{tag}} = (\mathsf{ct}_i^{\mathsf{w}}, \mathsf{ct}_i^{\mathsf{als\text{-}ip}}) \leftarrow \mathcal{E}.\mathsf{Enc}(\mathsf{ek}_i, \mathbf{x}_i, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$$

$$\text{where } \mathsf{ct}_i^{\mathsf{w}} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{Enc}(\mathsf{ek}_i^{\mathsf{w}}, \mathsf{tag}, \boxed{\mathbf{x}_i - r_i \cdot \mathbf{v}_i}, \mathsf{ac\text{-}ct}_i), \text{ and } \boxed{\mathsf{ct}_i^{\mathsf{als\text{-}ip}}} \leftarrow \mathcal{E}^{\mathsf{als\text{-}ip}}.\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{als\text{-}ip}}, \boxed{r_i \cdot \mathbf{t}_i})$$

$$\text{with: } \mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}} = (\mathsf{dk}^{\mathsf{w}}_{\mathbf{Y},\mathsf{ac\text{-}k}}, (\mathsf{dk}_i^{\mathsf{als\text{-}ip}})_i) \leftarrow \mathcal{E}.\mathsf{KeyExtract}(\mathsf{msk}, \mathbf{Y}, \mathsf{ac\text{-}k})$$

$$\text{where } \mathsf{dk}^{\mathsf{w}}_{\mathbf{Y},\mathsf{ac\text{-}k}} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{Extract}(\mathsf{msk}^{\mathsf{w}}, \mathbf{Y}, \mathsf{ac\text{-}k}) \text{ and } \boxed{\mathsf{dk}_i^{\mathsf{als\text{-}ip}}} \leftarrow \mathcal{E}^{\mathsf{als\text{-}ip}}.\mathsf{Extract}\left(\mathsf{msk}_i^{\mathsf{als\text{-}ip}}, \boxed{(\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}_i}\right)$$

where as previously mentioned, the function is written $\mathbf{Y} = [\mathbf{y}_1 \| \cdots \| \mathbf{y}_n]$.

The *correctness* follows that of the single-client IPFE scheme $\mathcal{E}^{\mathsf{als\text{-}ip}}$, and the correctness of $\mathcal{E}^{\mathsf{w}}$ notably with the preserved structural properties. The full proof for *IND-CPA security* under the strong admissibility condition is given with respect to Theorem 6. We highlight the main points below. Let $b \xleftarrow{\$} \{0,1\}$ denote the challenge bit,

- Focusing on lifting **W.1** to **S.1**, the difficult case is when an adversary corrupts i and queries $\mathbf{x}_i^{(0)} \neq \mathbf{x}_i^{(1)}$ to **LoR**.
- At an intuition level, the challenge at i contains $\mathsf{ct}_{i,\mathsf{tag}}^{(b)} = (\mathsf{ct}_i^{(\mathsf{w},b)}, \mathsf{ct}_i^{\mathsf{als\text{-}ip}})$ and $\mathsf{ct}_i^{(\mathsf{w},b)} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{Enc}(\mathsf{ek}_i^{\mathsf{w}}, \mathsf{tag}, \boxed{\mathbf{x}_i^{(b)} - r_i \cdot \mathbf{v}_i}, \mathsf{ac\text{-}ct}_i)$. This ensures that the difference $\Delta\mathbf{x} \coloneqq \mathbf{x}_i^{(b)} - \mathbf{x}_i^{(1)}$[12] is hidden in the randomness $r_i \cdot \mathbf{v}_i$. Under even partial decryption, this leaks $\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i \rangle - r_i \cdot \langle \mathbf{v}_i, \mathbf{y}_i \rangle - \underbrace{\langle \Delta\mathbf{x}_i, \mathbf{y}_i \rangle}_{=0 \text{ thanks to } \mathbf{S.1}}$ making the quantity independent from b. This step is formalized using the two-step decryption property under IND-CPA of $\mathcal{E}^{\mathsf{w}}$.
- For the above argument to work, we need to ensure that the IPFE layer *leaks nothing but the inner product evaluation*, which is solely needed for error-correcting the two-step decryption. The simulator of $\mathcal{E}^{\mathsf{als\text{-}ip}}$ is used to simulate $\mathsf{Sim}^{\mathsf{als\text{-}ip}}.\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{sim\text{-}als}}, \boxed{r_i \cdot \mathbf{t}_i})$, knowing the error-correcting values expected at the *corrupted client* i is $\langle r_i \cdot \mathbf{t}_i, (\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}_i \rangle$ while leaking nothing else. This leads to an exigence of *simulation-based* security for the IPFE scheme $\mathcal{E}^{\mathsf{als\text{-}ip}}$, else an indistinguishability-based cannot guarantee no information on $\boxed{r_i \cdot \mathbf{t}_i}$ is exposed, conditioned on the value $\langle r_i \cdot \mathbf{t}_i, (\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}_i \rangle$ where the adversary knows $\boxed{(\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}_i}$ due to *corruption*.
- We note that the private-repetitions of $\mathbf{x}_i$ at (honest) client i is preserved thanks to r_i freshly independently sampled at random for each encryption. Moreover *dynamic corruption* is handled thanks to resilience of $\mathcal{E}^{\mathsf{w}}$ to *dynamic* corruption of its $\mathsf{ek}_i^{\mathsf{w}}$, meanwhile the encryption of $\mathcal{E}^{\mathsf{als\text{-}ip}}$ is *public-key* and not affected by the corruption of multi-clients.
- Last but not least, when the functional keys does not decrypt the challenge ciphertexts, the admissibility condition **S.2** over the sum of honest i is irrelevant, corrupted i is already treated above, and the simulation security of $\mathcal{E}^{\mathsf{als\text{-}ip}}$ also extends to this case, as the adversary can *always decrypt* this layer of IPFE $\mathcal{E}^{\mathsf{als\text{-}ip}}$ (which emphasizes once more the importance of the simulation-based security). The latter guarantees that only values of $\langle r_i \cdot \mathbf{t}_i, (\frac{\mathbf{v}_i[k]}{\mathbf{t}_i[k]})_{k=1}^N \circ \mathbf{y}_i \rangle$ are disposed, which is independent from $\mathbf{x}_i^{(b)}$. The rest is reduced to the security of $\mathcal{E}^{\mathsf{w}}$ in case $\mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}}^{\mathsf{w}}$ does not decrypt the ensemble $(\mathsf{ct}_i^{\mathsf{w}})_i$.

In the end, the details of our transformation are given in Section 6. By combining with our MCFE from **Step 1** from LWE, the simulation secure IPFE scheme $\mathcal{E}^{\mathsf{als\text{-}ip}}$ from LWE in [ALS16, ALMT20], the LWE-based PRFs from [BPR12, BGM+16], we arrive at the first LWE-based MCFE for the class of inner products with access control that satisfies the *strong* admissibility condition.

1.3 Organization

In Section 2, we present the preliminaries required for this work. Next, in Section 3, we have the traitor tracing definitions, with two different variants for embedded identity tracing (EITT): indexed EITT and bounded EITT. In Section 4, we extend the definition of private linear broadcast encryption to the multi-client functional encryption setting (PL-MCFE) to give q-query PL-MCFE definitions and show how to use PL-MCFE to construct traitor tracing schemes. In Section 5, we generalize the transformation from indexed EITT to bounded EITT of [GKW19] to the multi-client setting. In Section 6, we present our compiler to achieve strong admissible security from weak admissible security for AB-MCFE. By combining this compiler with our AB-MCFE construction from Section 8, we derive the PL-MCFE construction from mixed FE and AB-MCFE, as detailed in Section 7.

[12] This is $\mathbf{0}$ when $b = 1$, possibly non-zero otherwise.

2 Preliminaries

Notation. We denote the security parameter by λ. A polynomial time algorithm A runs in time polynomial in the (implicit) security parameter λ. We denote "probabilistic polynomial time" by PPT. A function $f(\lambda)$ is negligible in λ if it is $\mathcal{O}(\lambda^{-c})$ for every $c \in \mathbb{N}$. We write $f = \mathsf{negl}(\lambda)$ for short. Similarly, we write $f = \mathsf{poly}(\lambda)$ if $f(\lambda)$ is a polynomial with variable λ. If D is a probability distribution, $x \leftarrow D$ means that x is sampled from D and if S is a set, $x \leftarrow S$ means that x is sampled uniformly and independently at random from S. We also write $|S|$ for the cardinality of set S. We denote with $[n]$ the set $\{1, \ldots, n\}$ for $n \in \mathbb{N}$. We use bold font for (row) vectors, and capitals for matrices.

2.1 Multi-client Functional Encryption with Fine-grained Access Control

Definition 1 (Attribute-based Functionality). *The function class $\mathcal{F} \times \mathsf{AC\text{-}K}$ for* functional evaluation with fine-grained access control *is described below:*

- *The function class $\mathcal{F} \times \mathsf{AC\text{-}K}$ contains $(F_\lambda, \mathsf{ac\text{-}k})$ having public attributes $(\mathsf{AC\text{-}Ct}_i)_{i\in[n]}$.*
- *The function F_λ has domain $\mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n}$ and range $\mathcal{R}_\lambda$.*
- *The public attributes of each i come from $\mathsf{Tag} \times \mathsf{AC\text{-}Ct}_i$ for some set $\mathsf{AC\text{-}Ct}_i$.*
- *The access control is defined via a relation $\mathsf{Rel} : \mathsf{AC\text{-}K} \times \mathsf{AC\text{-}Ct}_1 \times \cdots \times \mathsf{AC\text{-}Ct}_n \to \{0,1\}$, for some set $\mathsf{AC\text{-}K}$.*

The concrete evaluation is as follows:

$$(F_\lambda, \mathsf{ac\text{-}k}) \text{ evaluates on } ((x_i, \mathsf{ac\text{-}ct}_i)_{i=1}^n, \mathsf{ac\text{-}k}) = \begin{cases} F_\lambda(x_1, \ldots, x_n) & \text{if } \mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{ac\text{-}ct}_i)_i) = 1 \\ \bot & \text{otherwise} \end{cases}.$$

In our ABMCFE, a plaintext for client i consists of $x_i \in \mathcal{D}_{\lambda,i}$, where $\mathcal{D}_{\lambda,i}$ denotes the domain from which each client i gets their inputs, together with public tags and attributes form $\mathsf{Tag} \times \mathsf{AC\text{-}Ct}_i$ for some set $\mathsf{AC\text{-}Ct}_i$ and a tag space $\mathsf{Tag} = \{0,1\}^{poly(\lambda)}$. The corresponding ciphertexts can be decrypted to $F_\lambda(x)$ using the functional key $\mathsf{sk}_{F_\lambda,\mathsf{ac\text{-}k}}$ for $\mathsf{ac\text{-}k} \in \mathsf{AC\text{-}K}$ if and only if that ciphertexts contain the same tag from Tag *and* $\mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{ac\text{-}ct}_i)_i) = 1$.

In our FE with access control in this paper, we instantiate the computing class by $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ that is defined in Definition 11. The key-policy via $\mathsf{AC\text{-}K}$ is modeled by bounded-depth circuits as in [BGG+14].

Definition 2 (Multi-client functional encryption with fine-grained access control). *A* multi-client functional encryption (MCFE) scheme with fine-grained access control *for the function class $\mathcal{F} \times \mathsf{AC\text{-}K}$ consists of four algorithms* $(\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$*:*

$\mathsf{Setup}(1^\lambda, 1^k)$**:** *Given as input a security parameter λ, output a master secret key msk and $k = k(\lambda)$ encryption keys $(\mathsf{ek}_i)_{i\in[k]}$ where $k : \mathbb{N} \to \mathbb{N}$ is a function.*

$\mathsf{Extract}(\mathsf{msk}, (F_\lambda, \mathsf{ac\text{-}k}))$**:** *Given a circuit policy $\mathsf{ac\text{-}k} \in \mathsf{AC\text{-}K}$, a function description $F_\lambda \in \mathcal{F}$, and the master secret key msk, output a decryption key $\mathsf{dk}_{F_\lambda,\mathsf{ac\text{-}k}}$.*

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

$\mathsf{Dec}(\mathsf{dk}_{F_\lambda,\mathsf{ac\text{-}k}}, \mathbf{c})$**:** *Given the decryption key $\mathsf{dk}_{F_\lambda,\mathsf{ac\text{-}k}}$ and a vector of ciphertexts $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i}, \mathsf{tag})_i$ of length k, output an element in $\mathcal{R}_\lambda$ or an invalid symbol $\bot$.*

Correctness. For sufficiently large $\lambda \in \mathbb{N}$, for all $(\mathsf{msk}, (\mathsf{ek}_i)_{i\in[k]}) \leftarrow \mathsf{Setup}(1^\lambda)$, $(F_\lambda, \mathsf{ac\text{-}k}) \in \mathcal{F} \times \mathsf{AC\text{-}K}$ and $\mathsf{dk}_{F_\lambda,\mathsf{ac\text{-}k}} \leftarrow \mathsf{Extract}(\mathsf{msk}, F_\lambda, \mathsf{ac\text{-}k})$, for all tag and $(\mathsf{tag}_i)_i$ satisfying

$$\mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{tag}_i)_i) = 1$$

for all $(x_i)_{i\in[k]} \in \mathcal{D}_{\lambda,1} \times \cdots \times \in \mathcal{D}_{\lambda,k}$, if $F_\lambda(x_1,\ldots,x_k) \neq \perp$, the following holds with overwhelming probability:

$$\mathsf{Dec}\left(\mathsf{dk}_{F_\lambda,\mathsf{ac\text{-}k}}, (\mathsf{Enc}(\mathsf{ek}_i,\mathsf{tag},x_i,\mathsf{tag}_i))_{i\in[k]}\right) = F_\lambda(x_1,\ldots,x_k)$$

where $F_\lambda : \mathcal{D}_\lambda^k \to \mathcal{R}_\lambda$ and the probability is taken over the coins of algorithm.

Security. We follow the approach in the work by Chotard *et al.* [CDG+18a] so as to define the security game with oracles **Initialize**, **Corrupt**, **LoR**, **Enc**, **Extract**, and **Finalize**. We need to exclude trivial attacks that can be mounted in the security experiment. Those restrictions are encompassed in the notion of *admissibility*, which is recently extended in [NPP23] from similar notions in the works of [CDG+18a, CDSG+20].

Definition 3 (Weakly admissible adversaries without attribute repetitions). *Let $\mathcal{A}$ be a* PPT *adversary and let $\mathcal{E} = (\mathsf{Setup},\mathsf{Extract},\mathsf{Enc},\mathsf{Dec})$ be an MCFE scheme with fine-grained access control for the functionality class $\mathcal{F} \times \mathsf{AC\text{-}K}$. In the security game given in Figure 1 for $\mathcal{A}$ considering $\mathcal{E}$, let the sets $(\mathcal{C},\mathcal{Q},\mathcal{H})$ be the sets of corrupted clients, functional key queries, and honest clients, in that order. We say that $\mathcal{A}$* is NOT admissible w.r.t *$(\mathcal{C},\mathcal{Q},\mathcal{H})$ if any of the following conditions holds:*

1. *There exists $i \in \mathcal{C}$ such that $x_i^{(0)} \neq x_i^{(1)}$.*
2. *There exists $(\mathsf{tag},\mathsf{ac\text{-}ct}_i)$ for $i \in [k]$, a function $F \in \mathcal{F}$, and $\mathsf{ac\text{-}k} \in \mathsf{AC\text{-}K}$ such that*
   - *We have $\mathsf{Rel}(\mathsf{ac\text{-}k},(\mathsf{ac\text{-}ct}_i)_i) = 1$ and $(F,\mathsf{ac\text{-}k}) \in \mathcal{Q}$.*
   - *For all $i \in \mathcal{H}$, there exists a query $(i, x_i^{(0)}, x_i^{(1)}, \mathsf{tag}, \mathsf{ac\text{-}ct}_i)$ to* **LoR** *for $(x_i^{(0)}, x_i^{(1)})$.*
   - *For all $i \in \mathcal{C}$, it holds that $x_i^{(0)} = x_i^{(1)}$.*
   - *It holds that $F((x_i^{(0)})_{i\in[k]}) \neq F((x_i^{(1)})_{i\in[k]})$.*

Otherwise, we say that $\mathcal{A}$ is admissible *w.r.t $(\mathcal{C},\mathcal{Q},\mathcal{H})$.*

Definition 4 (Strong admissibility condition [NPP23]). *Let $\mathcal{A}$ be a* PPT *adversary and let $\mathcal{E} = (\mathsf{Setup},\mathsf{Extract},\mathsf{Enc},\mathsf{Dec})$ be an* MCFE scheme with fine-grained access control *for the functionality class $\mathcal{F} \times \mathsf{AC\text{-}K}$. In the security game given in Figure 1 for $\mathcal{A}$ considering $\mathcal{E}$, let the sets $(\mathcal{C},\mathcal{Q},\mathcal{H})$ be the sets of corrupted clients, functional key queries, and honest clients, in that order. We say that $\mathcal{A}$* is NOT admissible *w.r.t $(\mathcal{C},\mathcal{Q},\mathcal{H})$ if any of the following conditions holds:*

There exist $\mathsf{tag} \in \mathsf{Tag}$, a function $(F,\mathsf{ac\text{-}k}) \in \mathcal{Q}$ is queried to **Extract**, *a set of $2n$ challenges $(x_i^{(0)}, x_i^{(1)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i^{(chal)}))_{i\in[n]}$ are queried to* **LoR**, *with public inputs $\mathsf{ac\text{-}ct}_i^{(chal)} \in \mathsf{AC\text{-}Ct}_{\lambda,i}$, a pair $(\mathbf{t}^{(0)},\mathbf{t}^{(1)},\mathbf{v}^{(chal)})$ so that for $b \in \{0,1\}$, $\forall\, i \in \mathcal{H} : \mathbf{t}^{(b)}[i] = x_i^{(b)}$ and $\mathbf{v}^{(chal)}[i] = \mathsf{ac\text{-}ct}_i^{(chal)}$, and*

- *The policy passes*[13]*: $\mathsf{Rel}(\mathsf{ac\text{-}k},\mathbf{v}^{(chal)}) = 1$.*
- (Private-inputs only repetitions) *For any $i \in [n]$, there exists a unique query of the form $(x_i^{(0)}, x_i^{(1)}, (\mathsf{tag},*))$.*
- *The function evaluation differs:*

$$F(\mathbf{t}^{(0)}) \neq F(\mathbf{t}^{(1)}) \quad . \tag{2}$$

Otherwise, we say that $\mathcal{A}$ is admissible *w.r.t $(\mathcal{C},\mathcal{Q},\mathcal{H})$.*

Checking Admissibility. For all concrete classes that are considered in this work, the admissibility condition can be checked in polynomial-time at the finalisation of the security game, after receiving the guess $b' \in \{0,1\}$ of the adversary for the challenge bit $b \xleftarrow{\$} \{0,1\}$. Moreover, following [NPP23, NPP25], the admissibility translated for all classes in this work is optimal, that is, they cannot be relaxed further and the implicate security notion is the best we can hope for. In weaker notions, the checks are still taken into account, *e.g.,* for *static corruption* security, even after announcing the set of corrupted clients, the adversary is still allowed querying on these corrupted slots and in the end those queries are checked against Condition 2. Furthermore, in the foregoing case of static corruption, we implicitly assume that for corrupted slots the admissibility condition holds for self-crafted ciphertexts by the adversary.

[13] This is up to attributes replacement in the corrupted slots $i \in \mathcal{C}$, therefore we only required $\mathbf{v}^{(chal)}$ to coincide with only with the *honest* attributes $(\mathsf{ac\text{-}ct}_i^{(chal)})_{i\in\mathcal{H}}$ and leave free the *corrupted* part.

Remark 1 (Strong Admissibility for Inner Products with Access Control). The *strong* admissibility for $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} \times \mathsf{AC\text{-}K}$, followed the work of [NPP23] and deduced from Definition 4, is recalled below:

1. For all vectors $(\mathbf{x}_i^{(0,j_i)}, \mathbf{x}_i^{(1,j_i)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all $((\mathbf{y}_i)_{i\in[n]}, \mathsf{ac\text{-}k}) \in \mathcal{Q}$, let $\mathcal{H}$ be the set of honest clients and $b \xleftarrow{\$} \{0,1\}$ be the challenge bit. Then fof any $j_i \in [J_i]$, if $\mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{ac\text{-}ct}_i)_i) = 1$ then: $\sum_{i\in\mathcal{H}} \langle \mathbf{x}_i^{(b,j_i)} - \mathbf{x}_i^{(1,j_i)}, \mathbf{y}_i \rangle = 0$. This implies $\langle \mathbf{x}_i^{(b,j_i)} - \mathbf{x}_i^{(1,j_i)}, \mathbf{y}_i \rangle$ is constant for any $j_i \in [J_i]$. We recall that we are in the *private-inputs only repetitions* and therefore there are no repetitions over $(\mathsf{tag}, \mathsf{ac\text{-}ct}_i)$.
2. For all vectors $(\mathbf{x}_i^{(0,j_i)}, \mathbf{x}_i^{(1,j_i)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all $((\mathbf{y}_i)_{i\in[n]}, \mathsf{ac\text{-}k}) \in \mathcal{Q}$. Let $\mathcal{C} := [n] \setminus \mathcal{H}$ be the set of corrupted clients. Then, for all $i \in \mathcal{C}$, all $j_i \in [J]$: $\langle \mathbf{x}_i^{(b,j_i)} - \mathbf{x}_i^{(1,j_i)}, \mathbf{y}_i \rangle = 0$.

We recall that these conditions are for the *one-challenge, complete, with repetitions on private inputs* case and are checked in **Finalise** procedure at the end of the security experiment. Particularly, condition 2 is checked for all corrupted clients $i \in \mathcal{C}$ and all $j_i \in [J]$, given any queries that are made to the oracle **LoR** for $i \in \mathcal{C}$ by the adversary[14]. Finally, condition 2 does *not* need to cover private inputs of corrupted $i \in \mathcal{C}$ that are not queried to the oracle **LoR** because there exists no challenge bit b in those self-crafted ciphertexts $\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathbf{z}_i, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$. Decrypting $\mathsf{ct}_{\mathsf{tag},i}$ jointly with others challenge ciphertexts $\mathsf{ct}^{(b)}_{\mathsf{tag},j\neq i}$ under some key $\mathsf{dk}_{\mathsf{ac\text{-}k},(\mathbf{y}_i)_{i\in[n]}}$ always gives the same i-th component $\langle \mathbf{z}_i, \mathbf{y}_i \rangle$ regardless of b.

Remark 2 (Weaker Admissibility for Inner Products with Access Control). On the other hand, below is the weaker admissibility condition adm in Definition 3 for $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} \times \mathsf{AC\text{-}K}$, which was used in [NPP22] and in this work is first proved for our construction in Section 8 before passing via the transformationin Section 6 to achieve strong admissibility *as per* Remark 1:

1. For all vectors $(\mathbf{x}_i^{(0,j_i)}, \mathbf{x}_i^{(1,j_i)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all $((\mathbf{y}_i)_{i\in[n]}, \mathsf{ac\text{-}k}) \in \mathcal{Q}$, let $\mathcal{H}$ be the set of honest clients and $b \xleftarrow{\$} \{0,1\}$ be the challenge bit. Then fof any $j_i \in [J_i]$, if $\mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{ac\text{-}ct}_i)_i) = 1$ then: $\sum_{i\in\mathcal{H}} \langle \mathbf{x}_i^{(b,j_i)} - \mathbf{x}_i^{(1,j_i)}, \mathbf{y}_i \rangle = 0$. This implies $\langle \mathbf{x}_i^{(b,j_i)} - \mathbf{x}_i^{(1,j_i)}, \mathbf{y}_i \rangle$ is constant for any $j_i \in [J_i]$. We recall that we are in the *private-inputs only repetitions* and therefore there are no repetitions over $(\mathsf{tag}, \mathsf{ac\text{-}ct}_i)$.
2. For all vectors $(\mathbf{x}_i^{(0,j_i)}, \mathbf{x}_i^{(1,j_i)}, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ that is queried to **LoR**, for all $((\mathbf{y}_i)_{i\in[n]}, \mathsf{ac\text{-}k}) \in \mathcal{Q}$. Let $\mathcal{C} := [n] \setminus \mathcal{H}$ be the set of corrupted clients. Then, for all $i \in \mathcal{C}$, all $j_i \in [J]$: $\mathbf{x}_i^{(b,j_i)} = \mathbf{x}_i^{(1,j_i)}$.

Remark 3 (On Complete Challenge Messages). We remark that the challenges output by the adversary in the message hiding game of Definition 22 are required to be complete and contain both messages for each client $i \in [n]$. Since Definition 22 is a weaker notion of IND-CPA security, usual techniques from the literature in cases of concrete classes such as inner product functions [CDG+18b, CDSG+20] or attribute-based inner products [NPS25] can be used to allow *incomplete* challenge messages, *i.e.* lacking components potentially at some i. In this paper our concrete constructions will always use complete challenge messages for the sake of simplicity in proofs and main constructions' ideas.

Definition 5 (Strong IND-security for MCFE). *An MCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ *for the function class* $\mathcal{F} = \{F_\lambda\}_{\lambda\in\mathbb{N}}$ *is* xx-secure *if for all* PPT *adversaries* $\mathcal{A}$*, and for all sufficiently large* $\lambda \in \mathbb{N}$*, the following probability is negligible*

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

The game $\mathsf{Expr}^{\mathsf{xx}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ *is depicted in Figure 1. The security level indicator* xx *can be:* mc-ind-cpa *to indicate* IND-security *with adaptive challenges and dynamic corruption of* $(\mathsf{ek}_i)_{i=1}^n$*;* mc-sel-ind-cpa *to indicate* selective IND-security *with selective challenges constraint;* mc-sa-ind-cpa *to indicate* semi-adaptive IND-security *with adaptive challenges but selective key queries (*i.e.*, all key queries must be sent up front) contraint;* mc-ind-cpa-1chal *to indicate* one-time IND-security *with only one adaptive challenge tag and dynamic corruption of* ekey*. The probability is taken over the random coins of* $\mathcal{A}$ *and the algorithms.*

[14] This makes sense even in the case of *static corruption*, since we do *not* prohibit such queries even after the set $\mathcal{C}$ is fixed.

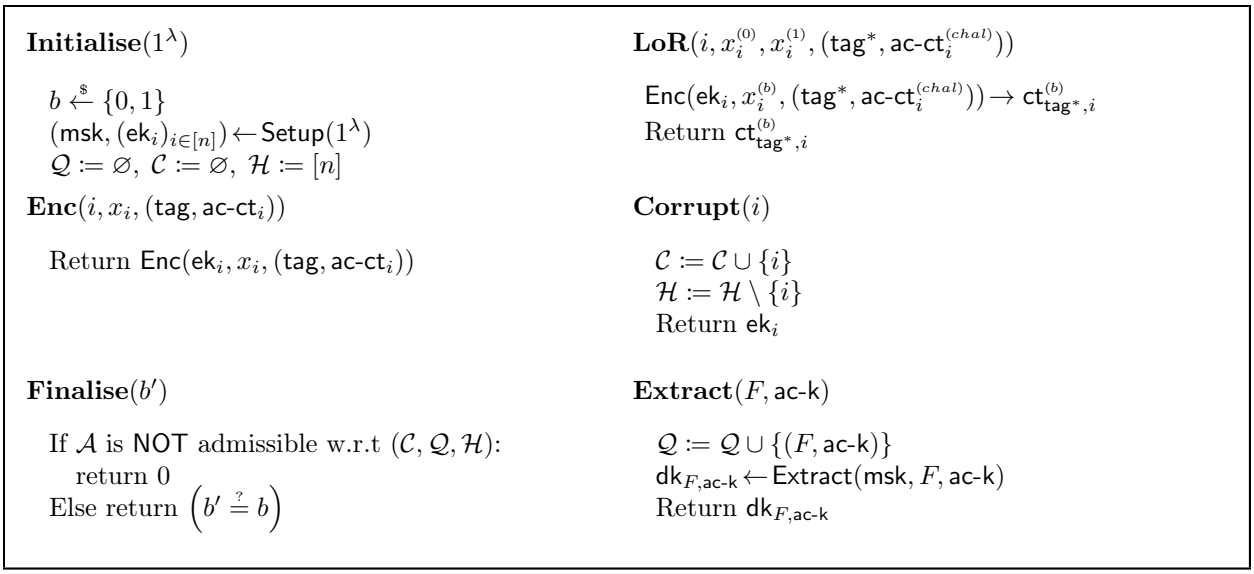

Fig. 1. The security games $\mathsf{Expr}^{\mathsf{mc\text{-}ind\text{-}cpa}}_{\mathcal{E},\mathcal{F},\mathcal{A}}(1^\lambda)$ for Definition 5. The *strong* admissibility condition is defined in Definition 4, the *weaker* admissibility condition is defined in Definition 3.

2.2 Mixed Functional Encryption

The syntax of a mixed functional encryption (mixed FE) scheme is defined as follows, first introduced in [GKW18]. The main idea behind mixed FE over functions with $\{0,1\}$-output is allowing at the same time both public-key and secret key encryption. The public-key encryption algorithm encrypts the all-1 function, while the secret-key encryption algorithm encryptions functions in $\mathcal{D}_\lambda \to \{0,1\}$ so as to be evaluated on functional keys that are associated to inputs $x \in \mathcal{D}_\lambda$.

Definition 6. *A mixed functional encryption scheme (mixedFE) for a function class $\mathcal{F} = \{f_\lambda : \mathcal{D}_\lambda^\kappa \to \{0,1\}\}_{\lambda\in\mathbb{N}}$, where $\kappa = \kappa(\lambda)$ is a polynomial in λ, consists of the following algorithms:*

$\mathsf{Setup}(1^\lambda, 1^\kappa)$**:** *Given as input a security parameter λ and the arity κ, output a master secret key* msk *and some public paramters* pp.
$\mathsf{Extract}(\mathsf{msk}, x)$**:** *Given the master secret key* msk *and an input $x \in \mathcal{D}_\lambda$, output a decryption key dk_x.*
$\mathsf{Enc}(\mathsf{pp})$**:** *Given the public parameters, output a ciphertext* ct.
$\mathsf{SKEnc}(\mathsf{msk}, f)$**:** *Given the the master secret key* msk *and a function $f : \mathcal{D}_\lambda^\kappa \to \{0,1\}$, output a ciphertext* ct.
$\mathsf{Dec}(\mathsf{dk}_x, \mathsf{ct})$**:** *Given the decryption key dk_x and a ciphertext* ct*, output a single bit.*

Correctness. For sufficiently large $\lambda \in \mathbb{N}$ and $\kappa : \mathbb{N} \to \mathbb{N}$ polynomial, for all $(\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa)$, $x \in \mathcal{D}_\lambda$ and $\mathsf{dk}_x \leftarrow \mathsf{Extract}(\mathsf{msk}, x)$, for all $f : \mathcal{D}_\lambda \to \{0,1\}$, the following holds with overwhelming probability:

$$\mathsf{Dec}\left(\mathsf{dk}_x, \mathsf{Enc}(\mathsf{pp})\right) = 1 \quad \text{and} \quad \mathsf{Dec}\left(\mathsf{dk}_x, \mathsf{SKEnc}(\mathsf{msk}, f)\right) = f(x).$$

where the probability is taken over the coins of algorithm.

Security. The formal definitions are given in [GKW18], we recall the informal ideas below. There are two main security properties: *(i - Function Indistinguishability)* for any PPT adversary, given any two functions $f_0, f_1 : \mathcal{D}_\lambda \to \{0,1\}$ that evaluates equally on all obtained keys dk_x of the adversary, *i.e.*, $f_0(x) = f_1(x)$ for all dk_x obtained from msk and $x \in \mathcal{D}_\lambda$, the adversary cannot distinguish between $\mathsf{SKEnc}(\mathsf{msk}, f_0)$ and $\mathsf{SKEnc}(\mathsf{msk}, f_1)$; *(ii - Accept Indistinguishability)* for any PPT adversary, given any function $f : \mathcal{D}_\lambda \to \{0,1\}$

that evaluates to 1 on all obtained keys dk_x of the adversary, *i.e.*, $f(x) = 1$ for all dk_x obtained from msk and $x \in \mathcal{D}_\lambda$, the adversary cannot distinguish between $\mathsf{Enc}(\mathsf{pp})$ and $\mathsf{SKEnc}(\mathsf{msk}, f)$. Formal definitions are recalled below.

Definition 7 (q-SKEnc Function Indistinguishability). *Let $\lambda \in \mathbb{N}$ and $q = q(\lambda)$ be some fixed polynomial in λ. A Mixed FE $\mathsf{mixedFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{SKEnc}, \mathsf{Dec})$ scheme is said to satisfy* adaptive q-SKEnc function indistinguishability *if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left| \Pr\left[\mathcal{A}^{\mathsf{Extract}(\mathsf{msk},\cdot),\mathsf{SKEnc}(\mathsf{msk},\cdot)}(\mathsf{ct}_b) = b \;\middle|\; \begin{array}{r} (1^\kappa) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa) \\ (f^{(0)}, f^{(1)}) \leftarrow \mathcal{A}^{\mathsf{Extract}(\mathsf{msk},\cdot),\mathsf{SKEnc}(\mathsf{msk},\cdot)}(\mathsf{pp}) \\ b \leftarrow\!\$\, \{0,1\}; \mathsf{ct}_b \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f^{(b)}) \end{array} \right] - \frac{1}{2} \right|$$

with the following oracle restrictions:

- SKEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SKEnc}(\mathsf{msk}, \cdot)$. *$\mathcal{A}$ can make at most q queries to* SKEnc.
- $\mathsf{Extract}$ *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{Extract}(\mathsf{msk}, \cdot)$. *Every query x to* $\mathsf{Extract}$ *must staisfy $f^{(0)}(x) = f^{(1)}(x)$.*

A Mixed FE $\mathsf{mixedFE}$ scheme is said to satisfy restricted q-SKEnc *function indistinguishability if we consider only adversaries that output $f^{(0)}, f^{(1)}$ before seeing the public parameters.*

Definition 8 (q-SKEnc Accept Indistinguishability). *Let $\lambda \in \mathbb{N}$ and $q = q(\lambda)$ be some fixed polynomial in λ. A Mixed FE $\mathsf{mixedFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{SKEnc}, \mathsf{Dec})$ scheme is said to satisfy* adaptive q-SKEnc accept indistinguishability *if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left| \Pr\left[\mathcal{A}^{\mathsf{Extract}(\mathsf{msk},\cdot),\mathsf{SKEnc}(\mathsf{msk},\cdot)}(\mathsf{ct}_b) = b \;\middle|\; \begin{array}{r} (1^\kappa) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{pp}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa) \\ f^{(*)} \leftarrow \mathcal{A}^{\mathsf{Extract}(\mathsf{msk},\cdot),\mathsf{SKEnc}(\mathsf{msk},\cdot)}(\mathsf{pp}) \\ b \leftarrow\!\$\, \{0,1\}; \\ \mathsf{ct}_1 \leftarrow \mathsf{SKEnc}(\mathsf{msk}, f^{(*)}); \mathsf{ct}_0 \leftarrow \mathsf{Enc}(\mathsf{pp}) \end{array} \right] - \frac{1}{2} \right|$$

with the following oracle restrictions:

- SKEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SKEnc}(\mathsf{msk}, \cdot)$. *$\mathcal{A}$ can make at most q queries to* SKEnc.
- $\mathsf{Extract}$ *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{Extract}(\mathsf{msk}, \cdot)$. *Every query x to* $\mathsf{Extract}$ *must staisfy $f^{(*)}(x) = 1$.*

A Mixed FE $\mathsf{mixedFE}$ scheme is said to satisfy restricted q-SKEnc accept indistinguishability *if we consider only adversaries that output $f^{(*)}$ before seeing the public parameters.*

2.3 Digital Signature

We recall the syntax and the *existential unforgeability under chosen-message attack* (EUF-CMA) security of a signature scheme.

Definition 9 (Signature Schemes). *Let $\lambda \in \mathbb{N}$ and $\mathcal{M} = \{0,1\}^*$ be the message space. A signature scheme $\mathsf{SS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ consists of three algorithms:*

$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$**:** *The setup algorithm takes as input the security parameter λ and outputs a signing key* sk *and a verification key* vk.

$\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$**:** *The signing algorithm takes as input the signing key* sk *and a message $m \in \mathcal{M}$ and outputs a signature σ.*

$b \leftarrow \mathsf{Verify}_{\mathsf{vk}}(m, \sigma)$**:** *The verification algorithm takes as input the verification key* vk*, a message $m \in \mathcal{M}$ and a signature σ and outputs a bit $b \in \{0,1\}$,* i.e. 1 *for "accept" and* 0 *for "reject".*

Correctness. A signature scheme is correct if for all $\lambda \in \mathbb{N}$, for all $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda)$, all $m \in \mathcal{M}$ and all $\sigma \leftarrow \mathsf{Sign}_{\mathsf{sk}}(m)$, it holds that $\mathsf{Verify}_{\mathsf{vk}}(m, \sigma) = 1$.

Security. A signature scheme is defined to be secure following the below notion of *existential unforgeability under chosen-message attack* (EUF-CMA) [PS00].

Definition 10 (EUF-CMA Security). *Let $\lambda \in \mathbb{N}$ and $\mathcal{M} = \{0,1\}^*$ be the message space. A signature scheme* $(\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ *is said to satisfy* adaptive EUF-CMA security *if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr\left[\mathsf{Verify}(\mathsf{vk}, \sigma^*) = 1 \;\middle|\; \begin{array}{r} (\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda) \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathsf{Sign}_{\mathsf{sk}}(\cdot)}(\mathsf{vk}) \end{array} \right]$$

with the following oracle:

- $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$*: this oracle has* sk *hardwired, and implements the algorithm* $\mathsf{Sign}(\mathsf{sk}, \cdot)$*. $\mathcal{A}$ it not allowed to query m^* to* $\mathsf{Sign}_{\mathsf{sk}}(\cdot)$*.*

3 Definitions: MCFE Traitor Tracing with Embedded Identities

Inner-product Functionality. The computation over clients' inputs this work is principally the sum of inner products *as per* the following.

Definition 11 (Inner-Product Functionality). *Let $\lambda \in \mathbb{N}$. Let $B = B(\lambda), n = n(\lambda), N_1(\lambda), \ldots, N_n(\lambda)$ be polynomials and $\mathcal{D}_i = [-B; B]^{N_i}$ for $i \in [n]$. We denote by $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} = \{f_{\mathbf{y}_1,\ldots,\mathbf{y}_n} : \mathbf{y}_1 \in \mathcal{D}_1, \ldots, \mathbf{y}_n \in \mathcal{D}_n\}$ the family of functions where $f_{\mathbf{y}_1,\ldots,\mathbf{y}_n} : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathbb{Z}_p$ is defined as $f_{\mathbf{y}_1,\ldots,\mathbf{y}_n}(\mathbf{x}_1, \ldots, \mathbf{x}_n) := \sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i \rangle$.*

3.1 Bounded Embedded-Identity Traitor Tracing

We will now present the syntax and definitions for general traitor tracing with embedded identities. In this work, we only consider the *bounded* collusion setting, where we have an a priori bound n_{bd} that is fixed during setup, and security is guaranteed only if the adversary gets at most n_{bd} secret keys.

Definition 12 ((Bounded) Embedded Identities Traceable Multi-Client Functional Encryption). *A* bounded keys, embedded-identities *tracing scheme $\mathcal{T}$ for a function class $\mathcal{F}$ consists of five algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *that are defined below:*

$(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{bd}})$**:** *Given as input the security parameter λ, the number of clients n, the identity space parameter κ, and bound on number of key queries n_{bd}, output a master secret key* msk*, public parameters* pp*, n encryption keys $(\mathsf{ek}_i)_{i\in[n]}$ and a tracing key* tk*.*

$\mathsf{dk}_{\mathsf{id},F_\lambda} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, F_\lambda)$**:** *Given the master secret key* msk*, an identity $\mathsf{id} \in \{0,1\}^\kappa$, a function description $F_\lambda \in \mathcal{F}$, output a decryption key $\mathsf{dk}_{\mathsf{id},F_\lambda}$.*

$\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$**:** *Given as inputs the public parameters* pp*, an encryption key ek_i, a message $\mathbf{x}_i \in \mathcal{D}_{\lambda,i}$, and a tag* tag*, output a ciphertext $\mathsf{ct}_{\mathsf{tag},i}$.*

$z \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F_\lambda}, \mathbf{c})$**:** *Given the public parameters* pp*, a decryption key dk_{F_λ} and a vector of ciphertexts $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i})_i$ of length n, output an element $z \in \mathcal{R}_\lambda \cup \{\perp\}$.*

$T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^y, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$**:** *Given oracle access to a program D, the tracing key* tk*, parameters y, a tag* tag *and two messages $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, output a set T of identities where $T \subseteq \{0,1\}^\kappa$.*
We call the tracing as public or private depending on whether tk *is equal to* pp *or it is kept secret.*

The correctness requirement and the IND-CPA security definition is identical to the one in Definition 5. We state here the tracing security.

- $1^\kappa, 1^{n_{\mathrm{bd}}} \leftarrow \mathcal{A}(1^\lambda)$
- $(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{bd}})$
- $(D, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \leftarrow \mathcal{A}^{\mathsf{Corrupt}(\cdot),\mathsf{KeyGen}(\cdot),\mathsf{Enc}(\cdot)}(\mathsf{pp})$
- Return $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$.

Let $S_{\mathcal{ID}}$ be the set of identities queried by $\mathcal{A}$ to $\mathsf{KeyGen}(\cdot)$.
Here, $\mathsf{Corrupt}(\cdot)$ is an oracle that has $(\mathsf{ek}_i)_{i\in[n]}$ hardwired, takes as input an index $i \in [n]$ and outputs ek_i; KeyGen is an oracle that has msk hardwired, takes as input an identity $\mathsf{id} \in \{0,1\}^\kappa$, a function description $F_\lambda \in \mathcal{F}$ and outputs $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, F_\lambda)$; Enc is an oracle that has $(\mathsf{ek}_i)_{i\in[n]}$ hardwired, takes as input a tag tag and a message $\mathbf{x}_i$ (we implicitly assume that the index of the encryption slot is embedded in x_i) and outputs $\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$.
Admissible tracing adversaries: $\mathcal{A}$ makes at most n_{bd} queries to KeyGen.

Fig. 2. Experiment $\mathsf{Expt}^{\mathsf{TT-emb}}$.

Definition 13 (Security of Tracing). *For any non-negligible function $\epsilon(\cdot)$, polynomial $p(\cdot)$, and all* PPT *adversary $\mathcal{A}$, consider the experiment $\mathsf{Expt}^{\mathsf{TT-emb}}_{\mathcal{T},\mathcal{A},\epsilon}(1^\lambda)$ defined in Fig. 2.*

Based on the above experiment in Fig. 2, we define the following events and corresponding probabilities (which are a function of λ, parameterized by $\mathcal{A}, \epsilon$):

- $\mathsf{GoodDec}$: $\Pr\left[\begin{array}{c} D\left((\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]}\right) = b \mid \\ b \leftarrow\!\!\$\ \{0,1\}, \mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i^{(b)}) \end{array}\right] \geq \frac{1}{2} + \epsilon(\lambda)$.
  $\mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{GoodDec}]$.
- $\mathsf{CorrectTr}$: $T \neq \varnothing \wedge T \subseteq S_{\mathcal{ID}}$.
  $\mathtt{Pr\text{-}CorrectTr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{CorrectTr}]$.
- $\mathsf{FalseTr}$: $T \not\subseteq S_{\mathcal{ID}}$.
  $\mathtt{Pr\text{-}FalseTr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{FalseTr}]$.

A traitor tracing scheme $\mathcal{T}$ is said to be secure if for every PPT admissible *adversary $\mathcal{A}$, polynomial $q(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exist negligible functions $\mathsf{negl}_1(\cdot), \mathsf{negl}_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > \frac{1}{q(\lambda)}$, it holds that $\mathtt{Pr\text{-}FalseTr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda)$ and $\mathtt{Pr\text{-}CorrectTr}_{\mathcal{A},\epsilon}(\lambda) \geq \mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda)$.*

We note that in the tracing security game of Figure 2, the adversary can extract separately secret information from both senders (via $\mathsf{Corrupt}$ to obtain secret encryption keys) and receivers (via KeyGen to obtain functional keys).

3.1.1 Impossibility

Next, we present justifications for our choices of modeling tracing security in the context of MCFE, that is, given the fact that our works is the first to embark on traceability with *multiple clients* encrypting using their *secret keys* so that their partial ciphertexts can be jointly decrypted under some functional key. In particular, we show an impossibility that there cannot exist a secure traitor tracing MCFE (following Definition 13) with a *public* tracing algorithm. We note that the definition we gave in Figure 2 only gives the tracer black-box access to the pirate decoder. Our impossibility will thus be proven for public black-box tracing, but as we will see later, our impossibility in fact applies to both black-box and non-black-box tracing.

In the follwing we refer to queries of functions that do *not* differ on the challenge messages $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ as *non-differentiating* queries. We require some structural properties of the function class, given *non-differentiating* functions and the number of clients $n \geq 2$, in the following sense:

Definition 14 (Slot-Differentiating Solvability). *Let $\lambda \in N$ and $\mathcal{F}$ be a function class parameterized by λ containing $F_\lambda : \mathcal{D}_1 \times \cdots \times \mathcal{D}_n \to \mathcal{R}_\lambda$, where $n = \mathsf{poly}(\lambda) \geq 2$. We say that $\mathcal{F}$ is* slot-differentiating solvable *if there exists $F_\lambda \in \mathcal{F}$, and fixed inputs $\mathbf{x}^{(0)}, \mathbf{x}^{(1)} \in \mathcal{D}_1 \times \cdots \times \mathcal{D}_n$, such that $F(\mathbf{x}^{(0)}) = F(\mathbf{x}^{(1)})$ and there exists a PPT algorithm that given $(F, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ outputs a set of indices $\varnothing \neq \mathcal{S} \subseteq [n]$ and $(\tilde{\mathbf{x}}_i^{(0)}, \tilde{\mathbf{x}}_i^{(1)})_{i \in \mathcal{S}}$ satisfying the following $F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) \neq F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right)$, where for $b \in \{0,1\}$*

$$\forall\, k \in [n] : \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(b)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(b)})_{i\in[n]\setminus\mathcal{S}}) \coloneqq \begin{cases} \tilde{\mathbf{x}}_k^{(b)} & \text{if } k \in \mathcal{S} \\ \mathbf{x}_k^{(b)} & \text{if } k \in [n] \setminus \mathcal{S} \end{cases}.$$

For the particular case of inner products that we consider in this work, $\mathcal{A}$ can even find $(F, \mathcal{S}, (\tilde{\mathbf{x}}_i^{(0)}, \tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}})$ on-the-fly by solving a linear system, for which we use inherently the condition that $n \geq 2$ as the number of clients/slots.

Theorem 1. *Let $\mathcal{T}$ be a traitor tracing* MCFE *scheme, for at least* 2 *clients and for a* slot-differentiating solvable *function class $\mathcal{F}$, so that $\mathcal{T}$ is* correct[15] *with* public black-box *tracing. If $\mathcal{T}$ is traceable (as per Definition 13), then $\mathcal{T}$ is* not *IND-CPA secure* as per *Definition 5.*

We first give an overview of the proof of Theorem 1. The goal is to show that if the tracing algorithm is public and black-box, then the adversary can break the IND-CPA security of the MCFE scheme. We make use of the two properties of the tracing algorithm:

- Since the tracing algorithm is public, the adversary will try to employ the tracing procedure to break the IND-CPA security. In particular, as our tracing security in Definition 13 identifies identities of the functional keys obtained to compose the priate decoder, the starting idea is to embed the challenge ciphertext as the identity in the function query to the tracing algorithm. This can be done as simple as identify the challenge ciphertext as some bitstring in the identity space, via some encoding: $\mathsf{bin} : \mathsf{CiphSpace} \to \{0,1\}^\kappa$.
- Next, we have to deal with the *strong admissibility* of our reduction, in terms of queries to the oracles in the IND-CPA game with repect to Definition 4. This is non-trivial since a *tracing* adversary is admissible as long as the number of key queries in its tracing game is bounded by n_{bd}.
- Our reduction to IND-CPA, acting as a tracing adversary, will first query a *non-differentiating* function F on some challenge $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, *i.e.* $F(\mathbf{x}^{(0)}) = F(\mathbf{x}^{(1)})$.
- Then, under the hypothesis that the number of clients $n \geq 2$, our reduction solves for a set of indices $\varnothing \neq \mathcal{S} \subseteq [n]$ and $(\tilde{\mathbf{x}}_i^{(0)}, \tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}$ such that $F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) \neq F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right)$, where for $b \in \{0,1\}$

  $$\forall\, k \in [n] : \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(b)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(b)})_{i\in[n]\setminus\mathcal{S}}) \coloneqq \begin{cases} \tilde{\mathbf{x}}_k^{(b)} \text{ if } k \in \mathcal{S} \\ \mathbf{x}_k^{(b)} \text{ if } k \in [n] \setminus \mathcal{S} \end{cases}.$$

  This uses the fact that the number of clients is at least 2 and the function class is *slot-differentiating solvable* (Definition 14).
- Then, the reduction runs the tracing algorithm, which has *black-box* access to a decoder that is embedded with the functional key for the function F under the identity that encodes the IND-CPA challenge ciphertext. The inputs to the tracing algorithm are $\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}}), \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})$, for $k \in [n]$ (defined following the slot-differentiating solvability). This can be freely specified by our reduction as the tracing algorithm is *public*.
- It can now be verified that the reduction is an *admissible tracing adversary* , the decoder that is black-box to the tracing algorithm is a good distinguisher *as per* the result from slot-differentiating solvability $F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) \neq F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right)$. At the same time, the reduction is also an *admissible IND-CPA adversary* (following Definition 4) thanks to the original condition $F(\mathbf{x}^{(0)}) = F(\mathbf{x}^{(1)})$.

[15] We do not require correctness to hold with probability 1, *i.e.* perfect, over the random coins of algorithms. Overwhelming probability for correctness suffices.

The remaining is running the tracing algorithm to output the set of identities, which include the identity that encodes the challenge ciphertext with high probability as $\mathcal{T}$ is tracing secure, then deciding the IND-CPA challenge bit (see the case-by-case decision in Equation (5)). As a final remark, we note that the fact that we pass a new set of inputs to the tracing algorithm, which are differentiated by F, implicitly demonstrates the ability to mix-and-match inputs in the context of MCFE and the later is not detected thanks to the black-box nature of the tracing algorithm.

We now give the formal proof.

Proof of Theorem 1. Let $\epsilon = 1/2$ and the number of keys $p(\lambda) = 1$. We describe $\mathcal{A}$ as an adversary against the IND-CPA security of $\mathcal{T}$ as follows:

- $\mathcal{A}$ is given the public parameters pp.
- $\mathcal{A}$ outputs two *distinct* messages $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ together with a tag tag, and receives a challenge ciphertext $\mathsf{ct}^{(b)}$.
- We denote by $\mathsf{bin} : \mathsf{CiphSpace} \to \{0,1\}^\kappa$ an encoding function that maps a ciphertext to a bitstring in the identity space $\mathcal{ID} = \{0,1\}^\kappa$ that acts as an identity[16]. $\mathcal{A}$ queries to $\mathsf{KeyGen}(\cdot)$ oracle a function F, setting identity $\mathsf{id}^{(b)} := \mathsf{bin}(\mathsf{ct}^{(b)}) \in \{0,1\}^\kappa$, where $F(\mathbf{x}^{(0)}) = F(\mathbf{x}^{(1)})$ but there exists $\varnothing \neq \mathcal{S} \subseteq [n]$, for which we make use of the hypothesis $n \geq 2$, and $(\tilde{\mathbf{x}}_i^{(0)}, \tilde{\mathbf{x}}_i^{(1)})_{i \in \mathcal{S}}$ such that

$$F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) \neq F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) \tag{3}$$

  where for $b \in \{0,1\}$

$$\forall\, k \in [n] : \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(b)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(b)})_{i\in[n]\setminus\mathcal{S}}) := \begin{cases} \tilde{\mathbf{x}}_k^{(b)} \text{ if } k \in \mathcal{S} \\ \mathbf{x}_k^{(b)} \text{ if } k \in [n] \setminus \mathcal{S} \end{cases} .$$

  The query of F is possible because $\mathcal{C} = \varnothing$ and thus there exists no deducible inputs to plug in the condition over $i \in \mathcal{C}$ of Definition 4, except the trivial ones $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$. This means the admissibility vacuously holds w.r.t $(F, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$. We denote by $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$ the received functional decryption key. We can suppose $\mathcal{A}$ is hardcoded with $(F, \varnothing \neq S \subseteq [n], \mathbf{x}^{(0)}, \mathbf{x}^{(1)}, (\tilde{\mathbf{x}}_i^{(0)}, \tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}})$ for the sake of an efficient adversary. The main idea behind embedding the challenge ciphertext as the identity $\mathsf{id}^{(b)}$ in the function query is to ensure that the tracing algorithm, supposedly secure, will identify $\mathsf{id}^{(b)}$ in the set T later on, so that the adversary can decide the bit b using this key $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$ and Equation (3).
- $\mathcal{A}$ receives a decryption key $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$. Because the tracing algorithm is *public*, $\mathcal{A}$ can run

$$T \leftarrow \mathsf{Trace}^D\left(\begin{array}{c} \mathsf{pp}, 1^{\frac{1}{\epsilon(\lambda)}}, p(\lambda), \mathsf{tag}, \\ \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n, \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n \end{array}\right), \tag{4}$$

  where D is the decryption algorithm hardwired with $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$, receiving ciphertext ct, decrypting with $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$ and outputs 1 if and only if

$$\mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F,\mathsf{id}^{(b)}}, \mathsf{ct}) = F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) .$$

  We emphasize the *public* tracing algorithm is run on a decoder from the challenge identity $\mathsf{id}^{(b)}$, but our adversary $\mathcal{A}$ deliberately puts

$$\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n, \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n$$

  as inputs to the tracer.

[16] This encoding bin can be as simple as the binary decomposition.

- If T contains an index that represents a ciphertext $\mathsf{bin}(\widetilde{\mathsf{ct}}) \in \{0,1\}^\kappa$, $\mathcal{A}$ decrypts using $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$ and defines

$$\begin{cases} \mathsf{result}_0 \coloneqq F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(0)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(0)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) - \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F,\mathsf{id}^{(b)}}, \widetilde{\mathsf{ct}}) \\ \mathsf{result}_1 \coloneqq F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(1)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) - \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F,\mathsf{id}^{(b)}}, \widetilde{\mathsf{ct}}) \end{cases} .$$

  Here the main idea is using the fact that *(i)* the identity $\mathsf{id}^{(b)}$ is the challenge ciphertext, *(ii)* $F(\mathbf{x}^{(0)}) = F(\mathbf{x}^{(1)})$, and (iii) the choices of $(F, \mathcal{S}, (\tilde{\mathbf{x}}_i^{(0)}, \tilde{\mathbf{x}}_i^{(1)})_{i\in\mathcal{S}})$ satisfying (3) to decide the bit b. $\mathcal{A}$ outputs

$$\begin{cases} \mathsf{guess} & \text{if } \exists\, \mathsf{guess} \in \{0,1\} \text{ s.t.} \\ & \quad \mathsf{result}_{\mathsf{guess}} = F\left(\Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(\mathsf{guess})})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(\mathsf{guess})})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n\right) - F(\mathbf{x}^{(\mathsf{guess})}) \,. \\ b' & \xleftarrow{\$} \{0,1\} \text{ otherwise} \end{cases} \tag{5}$$

  We recall that $\mathbf{x}^{\mathsf{guess}} \in \{\mathbf{x}^{(0)}, \mathbf{x}^{(1)}\}$ and $\mathbf{x}^{\mathsf{guess}}$ contains components $\mathbf{x}_i^{\mathsf{guess}}$ for $i \in [n]$.

For the ease of notation, we write

$$\widetilde{\mathbf{x}}^{(b)} \coloneqq \Delta_{\mathcal{S}}(k, (\tilde{\mathbf{x}}_i^{(b)})_{i\in\mathcal{S}}, (\mathbf{x}_i^{(b)})_{i\in[n]\setminus\mathcal{S}})_{i=1}^n$$

for $b \in \{0,1\}$ Since D is a black-box and hardwired with $\mathsf{dk}_{F,\mathsf{id}^{(b)}}$ differing on $(\widetilde{\mathbf{x}}^{(0)}, \widetilde{\mathbf{x}}^{(1)})$, which is implied by (3) and the fact that F is non-differentiating on $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, combining with perfect correctness of $\mathcal{T}$ implies that the event $\texttt{Pr-GoodDec}_{\mathcal{A},\epsilon,p}(\lambda) = 1 - \mathsf{negl}(\lambda)$ for some negligible function $\mathsf{negl}(\cdot)$. Because $\mathcal{T}$ is a secure traitor tracing MCFE scheme, there is a negligible function $\mathsf{negl}_2(\cdot)$ such that it holds

$$\begin{aligned} \texttt{Pr-CorrectTr}_{\mathcal{A},\epsilon,p}(\lambda) &\geq \texttt{Pr-GoodDec}_{\mathcal{A},\epsilon,p}(\lambda) - \mathsf{negl}_2(\lambda) \\ &= 1 - \mathsf{negl}(\lambda) - \mathsf{negl}_2(\lambda) \end{aligned} \tag{6}$$

and is overwhelming in λ. Additionally, there exists a negligible function $\mathsf{negl}_1(\cdot)$ such that

$$\texttt{Pr-FalseTr}_{\mathcal{A},\epsilon,p}(\lambda) \leq \mathsf{negl}_1(\lambda) \ .$$

Finally, we observe the facts that F is *differentiating* on $(\widetilde{\mathbf{x}}^{(0)}, \widetilde{\mathbf{x}}^{(1)})$ and the order of $\widetilde{\mathbf{x}}^{(b)}$ is *consistent* with the order of $\mathbf{x}^{(b)}$ for $b \in \{0,1\}$ when calling the tracing algorithm (4). Hence, whenever the tracing algorithm outputs a set T containing an id that represents the encoded challenge ciphertext, which happens with overwhelming probability thanks to (6), the differentiating functional key can decrypt and help $\mathcal{A}$ decide the bit b *as per* (5) (the cases are well-defined because of the design of F from (3)). □

3.2 Indexed Embedded-Identity Traitor Tracing

In this section, we will present the syntax and definitions for traitor tracing with embedded identities where the number of users is bounded, and the key generation is "indexed". The main motivation behind this *indexed* version is that we provide, in our *multi-client* setting, a *generic transformation* that turns an indexed EITT MCFE scheme into a (bounded) EITT MCFE (see an overview in Section 3.3).

Definition 15 (Indexed Embedded Identities Traceable Multi-Client Functional Encryption). *A* indexed keys, embedded-identities *tracing scheme* $\mathcal{T}$ *for a function class* $\mathcal{F}$ *consists of five algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Trace})$ *that are defined below:*

$(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}})$**:** *Given as input the security parameter* λ*, the number of clients* n*, the identity space parameter* κ *and index space* $[n_{\mathrm{indx}}]$*, output a master secret key* msk*, public parameters* pp*,* n *encryption keys* $(\mathsf{ek}_i)_{i\in[n]}$ *and a tracing key* tk*.*

$\mathsf{dk}_{(\mathsf{id},j),F_\lambda} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j, F_\lambda)$**:** *Given the master secret key* msk*, an identity* $\mathsf{id} \in \{0,1\}^\kappa$*, an index* $j \in [n_{\mathrm{indx}}]$*, and a function description* $F_\lambda \in \mathcal{F}$*, output a decryption key* $\mathsf{dk}_{(\mathsf{id},j),F_\lambda}$*.*

$\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$**:** *Given as inputs the public parameters* pp*, an encryption key* ek_i*, a message* $\mathbf{x}_i \in \mathcal{D}_{\lambda,i}$*, and a tag* tag*, output a ciphertext* $\mathsf{ct}_{\mathsf{tag},i}$*.*

$z \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F_\lambda}, \mathbf{c})$**:** *Given the public parameters* pp*, a decryption key* dk_{F_λ} *and a vector of ciphertexts* $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i})_i$ *of length* n*, output an element* $z \in \mathcal{R}_\lambda \cup \{\perp\}$*.*

$T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^y, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$**:** *Given oracle access to a program* D*, the tracing key* tk*, parameters* y*, a tag* tag *and two messages* $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$*, output a set* T *of identities where* $T \subseteq \{0,1\}^\kappa$*.*
We call the tracing as public or private depending on whether tk *is equal to* pp *or it is kept secret.*

The correctness requirement and the IND-CPA security definition is identical to the one in Definition 5. We state here the tracing security.

Definition 16 (Security of Tracing). *For any non-negligible function* $\epsilon(\cdot)$*, and all* PPT *adversary* $\mathcal{A}$*, consider the experiment* $\mathsf{Expt}^{\mathsf{TT-emb-index}}_{\mathcal{T},\mathcal{A},\epsilon}(1^\lambda)$ *defined in Fig. 3.*

- $1^\kappa, 1^{n_{\mathrm{indx}}} \leftarrow \mathcal{A}(1^\lambda)$
- $(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}})$
- $(D, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)}) \leftarrow \mathcal{A}^{\mathsf{Corrupt}(\cdot),\mathsf{KeyGen}(\cdot),\mathsf{Enc}(\cdot)}(\mathsf{pp})$
- Return $T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$.

Let $S_{\mathcal{ID}}$ be the set of identities queried by $\mathcal{A}$ to $\mathsf{KeyGen}(\cdot)$.
Here, $\mathsf{Corrupt}(\cdot)$ is an oracle that has $(\mathsf{ek}_i)_{i\in[n]}$ hardwired, takes as input an index $i \in [n]$ and outputs ek_i; KeyGen is an oracle that has msk hardwired, takes as input a pair $(j, \mathsf{id}) \in [n_{\mathrm{indx}}] \times \{0,1\}^\kappa$, a function description $F_\lambda \in \mathcal{F}$ and outputs $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j, F_\lambda)$ if index j is distinct from all previous queries made by $\mathcal{A}$, and outputs $\perp$ otherwise; Enc is an oracle that has $(\mathsf{ek}_i)_{i\in[n]}$ hardwired, takes as input a tag tag and a message $\mathbf{x}_i$ (we implicitly assume that the index of the encryption slot is embedded in x_i) and outputs $\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, x_i)$.
In other words, $\mathcal{A}$ *is allowed to make at most* one *query to* KeyGen *for each index* $j \in [n_{\mathrm{indx}}]$*.*

Fig. 3. Experiment $\mathsf{Expt}^{\mathsf{TT-emb-index}}$.

Based on the above experiment in Fig. 3, we define the following events and corresponding probabilities (which are a function of λ*, parameterized by* $\mathcal{A}, \epsilon$*):*

- GoodDec*:* $\Pr\left[\begin{array}{c} D\left((\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]}\right) = b \mid \\ b \leftarrow\$ \{0,1\}, \mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i^{(b)}) \end{array}\right] \geq \frac{1}{2} + \epsilon(\lambda)$.
  $\mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{GoodDec}]$.
- CorrectTr*:* $T \neq \varnothing \wedge T \subseteq S_{\mathcal{ID}}$.
  $\mathtt{Pr\text{-}CorrectTr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{CorrectTr}]$.
- FalseTr*:* $T \not\subseteq S_{\mathcal{ID}}$.
  $\mathtt{Pr\text{-}FalseTr}_{\mathcal{A},\epsilon}(\lambda) = \Pr[\mathsf{FalseTr}]$.

A scheme $\mathcal{T}$ *is said to be secure if for every* PPT *adversary* $\mathcal{A}$*, polynomial* $q(\cdot)$ *and non-negligible function* $\epsilon(\cdot)$*, there exist negligible functions* $\mathsf{negl}_1(\cdot), \mathsf{negl}_2(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *satisfying* $\epsilon(\lambda) > \frac{1}{q(\lambda)}$*, it holds that* $\mathtt{Pr\text{-}FalseTr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}_1(\lambda)$ *and* $\mathtt{Pr\text{-}CorrectTr}_{\mathcal{A},\epsilon}(\lambda) \geq \mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}_2(\lambda)$.

3.3 Overview: Framework for Embedded-Identity Traitor Tracing MCFE

Before moving on to other sections, we give an overview of the framework that involves our newly defined notions of bounded/indexed embedded-identity traitor tracing MCFE. In the setting of traitor tracing for

public key encryption (PKE), the work of [GKW18, GKW19] provides a framework to build *embedded-identity traitor tracing* (EITT) from *indexed* EITT. All this works and other follow-ups are in the setting of *single-client*, that is there is only one sender that encrypts the message, facing *multiple users* that can decrypt on keys embedded with identities. However, as it should be clear from the expository overview during the course of Section 1.2.1, the traitor tracing MCFE model we propose are justified and effectively in the *multi-client* setting. A constructive framework for our *embedded-identity traitor tracing MCFE* (EITT-MCFE) will be the first to tackle this context and face the following challenges:

- We choose to trace *privately* and preferably in a *black-box* manner (this is due to our impossibility Theorem 1).
- The complexity of *private-key with corruption* of multi-clients.
- The *fine-grained separate corruptions* of clients and functional keys under the necessity of *strong admissibility*.

Generalizing [GKW18] to TMCFE. We generalize the framework of [GKW18] to the setting of *multi-client* functional encryption, with traitor tracing capabilities facing *multi-sender* that can ask *bounded* number of functional keys. The final primitive is the *bounded* EITT-MCFE (Definition 12), for the class computing inner products $\sum_{i=1}^{n}\langle \mathbf{x}_i, \mathbf{y}_i\rangle$ between n clients. In particular, we develop on both definitional and constructive aspects for the technique of *private linear broadcast tracing with embedded identities* [BSW06, GKW19]. Our roadmap is as follows.

TMCFE.1 The above transformation reduces to the task of constructing indexed EITT for MCFE. To that end, we first introduce a new intermediate primitive which we call *embedded-identity private linear multi-client functional encryption* (EI-PLMCFE, Definition 17). As for all primitives we extend to the multi-client setting, the security notions for EI-PLMCFE in Section 4.1.1 all imbue the fine-grained corruption of multiple clients and functional keys, while the strong-amissibly IND-CPA is reflected in the *message-hiding* of EI-PLCMFE (Definition 22). Eventually, we give a *generic transformation* from EI-PLMCFE to indexed EITT for MCFE in Section 4.

TMCFE.2 Finally, we show *how to build generically* EI-PLMCFE from *Mixed Functional Encryption* (Mixed FE) (introduced in [GKW18]) and MCFE for inner products with access control with *strong* admissibility conditions (see Section 7). Both of them can be constructed based on standard LWE assumptions.

TMCFE.3 Finally, we provide, in our *multi-client* setting, a *generic transformation* that turns an indexed EITT MCFE scheme into a (bounded) EITT MCFE (see Section 5).

A corresponding overview with more details will also accompany in each of the referred sections.

4 Building Indexed EITT-MCFE

4.1 Embedded-Identity Private Linear MCFE

We develop a new notion of *embedded-identity private linear MCFE (EI-PLMCFE)*, which can be seen as an extension of classical embedded-identity private linear broadcast encryption introduced in [GKW19].

Definition 17. *An* embedded-identity private linear MCFE (EI-PLMCFE) $\mathsf{EIPLMCFE}$ *for a function class* $\mathcal{F}$ *and identity space* $\mathcal{ID} = \{0,1\}^{\kappa}$ *consists of five algorithms* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{SplEnc}, \mathsf{Dec})$ *that are defined below:*

$(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^{\lambda}, 1^{n}, 1^{\kappa}, n_{\mathrm{indx}})$**:** *Given as input the security parameter* λ*, the number of clients* n*, the identity space parameter* κ*, and the index space* $[n_{\mathrm{indx}}]$*, output a master secret key* msk*, public parameters* pp*, and* n *encryption keys* $(\mathsf{ek}_i)_{i\in[n]}$*.*

$\mathsf{dk}_{(\mathsf{id},j),F_\lambda} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j, F_\lambda)$**:** *Given as input the master secret key* msk*, an identity* $\mathsf{id} \in \mathcal{ID}$*, an index* $j \in [n_{\mathrm{indx}}]$*, and a function description* $F_\lambda \in \mathcal{F}$*, output a decryption key* $\mathsf{dk}_{(\mathsf{id},j),F_\lambda}$*.*

$\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)$**:** *Given as inputs the public parameters* pp*, an encryption key* ek_i*, a tag* tag*, and a message* $\mathbf{x}_i \in \mathcal{D}_{\lambda,i}$*, output a ciphertext* $\mathsf{ct}_{\mathsf{tag},i}$*.*

$\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \widehat{\mathsf{tag}} := (\mathsf{tag}, (j, \ell, b)), \mathbf{x}_i)$**:** *Given as inputs the master secrete key* msk*, a tag* tag*, a triple* $(j, \ell, b) \in [n_{\mathrm{indx}} + 1] \times ([\kappa] \cup \{\perp\}) \times \{0, 1\}$*, and a message* $\mathbf{x}_i \in \mathcal{D}_{\lambda,i}$*, and output a ciphertext* $\mathsf{ct}_{\mathsf{tag},i}$*. For simplicity, we will only write* $\mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, b), \mathbf{x}_i)$*.*

$z \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F_\lambda}, \mathbf{c})$**:** *Given the public parameters* pp*, a decryption key* dk_{F_λ} *and a vector of ciphertexts* $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i})_i$ *of length* n*, output an element* $z \in \mathcal{R}_\lambda \cup \{\perp\}$*.*

Remark 4. For convenience, we make the following convention: we use i to denote the clients' index (on ciphertexts), and j (e.g., $j, j', \ldots$) to denote users' index (on functional keys).

Correctness. An EI-PLMCFE scheme is said to be correct if for all $\lambda, n, n_{\mathrm{indx}}, \kappa \in \mathbb{N}$, an identity space $\mathcal{ID} = \{0,1\}^\kappa$, $F \in \mathcal{F}_\lambda : \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n} \to \mathcal{R}_\lambda$, $i \in [n], j \in [n_{\mathrm{indx}} + 1], j' \in [n_{\mathrm{indx}}], \mathsf{id} \in \mathcal{ID}, \ell \in [\kappa] \cup \{\perp\}, b \in \{0,1\}$, tag and $(\mathbf{x}_i)_{i\in[n]} \in \mathcal{D}_{\lambda,1} \times \cdots \times \mathcal{D}_{\lambda,n}$ the following probability:

$$\Pr\left[\begin{array}{c|r} \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F_\lambda}, (\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]}) & (\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}}) \\ = F(\mathbf{x}_1, \ldots, \mathbf{x}_n) & \mathsf{dk}_{(\mathsf{id},j),F_\lambda} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j, F_\lambda) \\ & (\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]} \leftarrow \{\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)\}_{i\in[n]} \end{array}\right]$$

is negligible in λ, and if $j' \geq j + 1$ or $(j, \ell) = (j', \perp)$ or $(j, \mathsf{id}_j) = (j', 1 - b)$ then

$$\Pr\left[\begin{array}{c|r} \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F_\lambda}, (\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]}) & (\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}}) \\ = F(\mathbf{x}_1, \ldots, \mathbf{x}_n) & \mathsf{dk}_{(\mathsf{id},j'),F_\lambda} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j', F_\lambda) \\ & (\mathsf{ct}_{\mathsf{tag},i})_{i\in[n]} \leftarrow \{\mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, b), \mathbf{x}_i)\}_{i\in[n]} \end{array}\right]$$

is negligible in λ, where the probabilities are taken over the random coins of algorithms.

4.1.1 q-Query EI-PLMCFE Security

In this section, we provide the formal security definitions for EI-PLMCFE.

Definition 18 (q-query Normal Hiding). *Let $\lambda \in \mathbb{N}$ and $q = q(\lambda)$ be some fixed polynomial in λ. An EI-PLMCFE scheme satisfies* q-query normal hiding *security if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{SplEnc}(\cdot)}(\mathbf{c}^{(b)}) = b \;\middle|\; \begin{array}{r} (1^\kappa, 1^{n_{\mathrm{indx}}}) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}}) \\ (\mathsf{tag}, \mathbf{x}_i) \leftarrow \mathcal{A}^{\mathsf{SplEnc}(\cdot),\mathsf{KeyGen}(\cdot)}_{\mathsf{Enc}(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{pp}) \\ b \leftarrow\$ \{0,1\} \\ \mathsf{ct}_i^{(0)} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i) \\ \mathsf{ct}_i^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (1, \perp, 0), \mathbf{x}_i) \\ \mathbf{c}^{(b)} \leftarrow \left(\mathsf{ct}_i^{(b)}\right)_{i\in[n]} \end{array}\right] - \frac{1}{2}\right|$$

with the following oracle restrictions:

- Enc *oracle: this oracle has* $(\mathsf{ek}_i)_{i\in[n]}$ *hardwired, and implements the algorithm* $\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \cdot, \cdot)$*.*
- $\mathsf{Corrupt}$ *oracle: this oracle has* $(\mathsf{ek}_i)_{i\in[n]}$ *hardwired, and takes as input a client index i and returns* ek_i*.*
- SplEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SplEnc}(\mathsf{msk}, \cdot, \cdot, \cdot)$*. $\mathcal{A}$ can make* at most q queries *to* SplEnc*. And for each query on (j, ℓ, b) the user-index j has to be* 1*.*
- KeyGen *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot, \cdot)$*. $\mathcal{A}$ can make* at most one query for each user-index $j \in [n_{\mathrm{indx}}]$*. That is, let $(\mathsf{id}_1, j_1), \ldots, (\mathsf{id}_k, j_k)$ denote all the key queries made by $\mathcal{A}$, then $j_a \neq j_b$ for all $a \neq b$.*

Definition 19 (q-query Index Hiding). *Let $\lambda \in \mathbb{N}$ and $q = q(\lambda)$ be some fixed polynomial in λ. An EI-PLMCFE scheme satisfies* q-query index hiding security *if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left| \Pr\left[\mathcal{A}^{\mathsf{SplEnc}(\cdot)}(\mathbf{c}^{(b)}) = b \;\middle|\; \begin{array}{r} (1^\kappa, 1^{n_{\mathrm{indx}}}, j^*) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}}) \\ (\mathsf{tag}, \mathbf{x}_i) \leftarrow \mathcal{A}^{\mathsf{SplEnc}(\cdot),\mathsf{KeyGen}(\cdot)}_{\mathsf{Enc}(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{pp}) \\ b \leftarrow\$ \{0,1\} \\ \mathsf{ct}^{(b)} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j^*+b, \perp, 0), \mathbf{x}_i) \\ \mathbf{c}^{(b)} \leftarrow \left(\mathsf{ct}_i^{(b)}\right)_{i\in[n]} \end{array} \right] - \frac{1}{2} \right|$$

with the following oracle restrictions:

- Enc *and* Corrupt *oracles are defined identically as in Definition 18.*
- SplEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SplEnc}(\mathsf{msk}, \cdot, \cdot, \cdot)$. *$\mathcal{A}$ can make* at most q queries *to* SplEnc. *And for each query on $((j, \ell, b))$ the user-index j has to be either j^* or $j^* + 1$.*
- KeyGen *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot, \cdot)$. *$\mathcal{A}$ can make* at most one query for each user-index $j \in [n_{\mathrm{indx}}]$, *and* no key query of the form (id, j^*). *That is, let $(\mathsf{id}_1, j_1), \ldots, (\mathsf{id}_k, j_k)$ denote all the key queries made by $\mathcal{A}$, then $j_a \neq j_b$ for all $a \neq b$. And, $j_a \neq j^*$ for all a.*

Definition 20 (q-query Upper Identity Hiding). *Let $\lambda \in \mathbb{N}$ and $q = q(\lambda)$ be some fixed polynomial in λ. An EI-PLMCFE scheme satisfies* q-query upper identity hiding security *if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left| \Pr\left[\mathcal{A}^{\mathsf{SplEnc}(\cdot)}(\mathbf{c}^{(b)}) = b \;\middle|\; \begin{array}{r} (1^\kappa, 1^{n_{\mathrm{indx}}}, j^*, \ell^*, b^*) \leftarrow \mathcal{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}}) \\ (\mathsf{tag}, \mathbf{x}_i) \leftarrow \mathcal{A}^{\mathsf{SplEnc}(\cdot),\mathsf{KeyGen}(\cdot)}_{\mathsf{Enc}(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{pp}) \\ b \leftarrow\$ \{0,1\} \\ \mathsf{ct}_i^{(b)} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j^*+1, \perp, 0), \mathbf{x}_i) \\ \mathsf{ct}_i^{(b)} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j^*, \ell^*, b^*), \mathbf{x}_i) \\ \mathbf{c}^{(b)} \leftarrow \left(\mathsf{ct}_i^{(b)}\right)_{i\in[n]} \end{array} \right] - \frac{1}{2} \right|$$

with the following oracle restrictions:

- Enc *and* Corrupt *oracles are defined identically as in Definition 18.*
- SplEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SplEnc}(\mathsf{msk}, \cdot, \cdot, \cdot)$. *$\mathcal{A}$ can make* at most q queries *to* SplEnc. *And for each query on $((j, \ell, b))$ the user-index j has to be either j^* or $j^* + 1$.*
- KeyGen *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot, \cdot)$. *$\mathcal{A}$ can make* at most one query for each user-index $j \in [n_{\mathrm{indx}}]$, *and* no key query of the form (id, j^*) *where $\mathsf{id}_{\ell^*} = 1 - b^*$. That is, let $(\mathsf{id}_1, j_1), \ldots, (\mathsf{id}_k, j_k)$ denote all the key queries made by $\mathcal{A}$, then $j_a \neq j_b$ for all $a \neq b$. And, $(\mathsf{id}_a)_{\ell^*} \neq 1 - b^*$ or $j_a \neq j^*$ for all a.*

Definition 21 (q-query Lower Identity Hiding). *Let $\lambda \in \mathbb{N}$ and $q = q(\lambda)$ be some fixed polynomial in λ. An EI-PLMCFE scheme satisfies* q-query lower identity hiding security *if for every stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{SplEnc}(\cdot)}(\mathbf{c}^{(b)})=b \;\middle|\; \begin{array}{r}(1^\kappa,1^{n_{\mathrm{indx}}},j^*,\ell^*,b^*)\leftarrow\mathcal{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{pp},(\mathsf{ek}_i)_{i\in[n]})\leftarrow\mathsf{Setup}(1^\lambda,1^n,1^\kappa,n_{\mathrm{indx}})\\ (\mathsf{tag},\mathbf{x}_i)\leftarrow\mathcal{A}^{\mathsf{SplEnc}(\cdot),\mathsf{KeyGen}(\cdot)}_{\mathsf{Enc}(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{pp})\\ b\leftarrow\$\,\{0,1\}\\ \mathsf{ct}_i^{(b)}\leftarrow\mathsf{SplEnc}(\mathsf{msk},\mathsf{tag},(j^*,\perp,0),\mathbf{x}_i)\\ \mathsf{ct}_i^{(b)}\leftarrow\mathsf{SplEnc}(\mathsf{msk},\mathsf{tag},(j^*,\ell^*,b^*),\mathbf{x}_i)\\ \mathbf{c}^{(b)}\leftarrow\left(\mathsf{ct}_i^{(b)}\right)_{i\in[n]}\end{array}\right]-\frac{1}{2}\right|$$

with the following oracle restrictions:

- Enc *and* Corrupt *oracles are defined identically as in Definition 18.*
- SplEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SplEnc}(\mathsf{msk},\cdot,\cdot,\cdot)$. *$\mathcal{A}$ can make* at most q queries to SplEnc. *And for each query on $((j,\ell,b))$ the user-index j has to be j^*.*
- KeyGen *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot)$. *$\mathcal{A}$ can make* at most one query for each user-index $j\in[n_{\mathrm{indx}}]$, *and* no key query of the form (id,j^*) *where* $\mathsf{id}_{\ell^*}=b^*$. *That is, let $(\mathsf{id}_1,j_1),\ldots,(\mathsf{id}_k,j_k)$ denote all the key queries made by $\mathcal{A}$, then $j_a\neq j_b$ for all $a\neq b$. And, $(\mathsf{id}_a)_{\ell^*}\neq b^*$ or $j_a\neq j^*$ for all a.*

The following Definition 22 defines the message hiding security for EI-PLMCFE, which intuitively captures a weaker version of IND-CPA security that we will base the tracing security on: for any PPT adversay, after constructing a decoder with oracle access to key-generation, encryption, as well as corruption oracles, on chosen tag and challenge messages, given the challenge ciphertext, this adversarial decoder is not able to distinguish the challenge bit. We remark that in [GKW19] the message hiding security is rather defined similarly to the full-fledged IND-CPA security of PKE schemes.

Definition 22 (q-query Message Hiding). *Let $\lambda\in\mathbb{N}$ and $q=q(\lambda)$ be some fixed polynomial in λ. An EI-PLMCFE scheme satisfies* adaptive q-query message hiding security *if for every* admissible *stateful* PPT *adversary $\mathcal{A}$, the following probability is negligible in λ:*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{SplEnc}(\cdot)}(\mathbf{c}^{(b)})=b \;\middle|\; \begin{array}{r}(1^\kappa,1^{n_{\mathrm{indx}}})\leftarrow\mathcal{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{pp},(\mathsf{ek}_i)_{i\in[n]})\leftarrow\mathsf{Setup}(1^\lambda,1^n,1^\kappa,n_{\mathrm{indx}})\\ \left(\mathsf{tag},\left(\mathbf{x}_i^{(0)},\mathbf{x}_i^{(1)}\right)_{i\in[n]}\right)\leftarrow\mathcal{A}^{\mathsf{SplEnc}(\cdot),\mathsf{KeyGen}(\cdot)}_{\mathsf{Enc}(\cdot),\mathsf{Corrupt}(\cdot)}(\mathsf{pp})\\ b\leftarrow\$\,\{0,1\}\\ \mathsf{ct}_i^{(b)}\leftarrow\mathsf{SplEnc}(\mathsf{msk},\mathsf{tag},(n_{\mathrm{indx}}+1,\perp,0),\mathbf{x}_i^{(b)})\\ \mathbf{c}^{(b)}:=\left(\mathsf{ct}_i^{(b)}\right)_{i=1}^n\end{array}\right]-\frac{1}{2}\right|$$

with the following oracle restrictions:

- Enc *and* Corrupt *oracles are defined identically as in Definition 18.*
- SplEnc *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{SplEnc}(\mathsf{msk},\cdot,\cdot,\cdot)$. *$\mathcal{A}$ can make* at most q queries *to* SplEnc. *And for each query on $((n_{\mathrm{indx}}+1,\ell,b))$ the user-index has to be $n_{\mathrm{indx}}+1$.*
- KeyGen *oracle: this oracle has* msk *hardwired, and implements the algorithm* $\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot)$. *$\mathcal{A}$ can make* at most one query for each user-index $j\in[n_{\mathrm{indx}}]$. *That is, let $(\mathsf{id}_1,j_1),\ldots,(\mathsf{id}_k,j_k)$ denote all the key queries made by $\mathcal{A}$, then $j_a\neq j_b$ for all $a\neq b$.*

Furthermore, the adversary is admissible if it satisfies Definition 23 for the specific function class computing inner products.

Definition 23 (Strong Admissibility for Inner Products). *Below we recall the* strong *admissibility condition (against* complete *queries) for MCFE for the function class that compute inner products (Definition 11) of $\langle\mathbf{X},\mathbf{Y}\rangle$ where $\mathbf{X}=\mathbf{x}_1\|\cdots\|\mathbf{x}_n$, and each ek_i is used to encryption a vector $\mathbf{x}_i$. We note that formal definition of this strong admissibility is given in Definition 4, which is inspired by the definition in [NPP23].*

1. *For all vectors* $(\mathbf{X}^{(0)}, \mathbf{X}^{(1)})$ *that is queried to* **LoR**, *for all* $(\mathsf{tag}, \mathbf{Y} = (\mathbf{y}_i)_i^n) \in \mathcal{Q}$, $\sum_{i \in \mathcal{H}} \langle \Delta\mathbf{x}_i, \mathbf{y}_i \rangle = 0$ *where* $\Delta\mathbf{x}_i = \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(b)}$, *where* $\mathcal{H} := \mathcal{H}_{\mathsf{ekey}}$, *for any* $b \in \{0,1\}$.
2. *For all vectors* $(\mathbf{X}^{(0)}, \mathbf{X}^{(1)})$ *that is queried to* **LoR**, *for all* $(\mathsf{tag}, \mathbf{Y} = (\mathbf{y}_i)_i^n) \in \mathcal{Q}$, *for all* $i \in \mathcal{C}_{\mathsf{ekey}}$, *we have* $\langle \Delta\mathbf{x}_i, \mathbf{y}_i \rangle = 0$.

We recall that the admissbility condition is checked at the finalisation of the security game, after receiving the guess $b' \in \{0,1\}$ of the adversary for the challenge bit $b \xleftarrow{\$} \{0,1\}$. The conditions are applied to all queries by the adversary during the game.

Remark 5 (Comparing with [GKW19]). The acute reader might notice that we do not allow the adversary's access to the KeyGen oracle after the challenge phase, while [GKW19] does allow. However, this oracle access is not necessary, in both our security proofs in Section 4.2.1 and those of [GKW19]. In fact, it is crucial for our construction as it only require the functional keys to be semi-adaptive, the security level of our building blocks in Section 6 are compatible.

4.2 Building Indexed EITT from EI-PLMCFE

Our construction of indexed EITT for MCFE from EI-PLMCFE (Item **TMCFE.1**) consists of our new integration of a tensoring technique for the *multi-client* setting, all along with non-interactive secret sharing of 0 among clients, see the main transformation in Section 4.2.1. This extends fully the works of [ABG19] and [LAKWH22] to the tracing of *multi-sender* setting. Starting from the case of *one* client encrypting a message vector $\mathbf{x}$, deryptable using a functional key for vector $\mathbf{y}$, in order to *index*-trace the users' decryption keys, our main idea is letting the indexed EITT-MCFE tensors the vectors $\mathbf{x}, \mathbf{y}$ with a pair of vectors $(\mathbf{a}, \mathbf{b})$ of appropriate length that, by properties of tensor products, $\langle \mathbf{x} \otimes \mathbf{a}, \mathbf{y} \otimes \mathbf{b} \rangle = \langle \mathbf{x}, \mathbf{y} \rangle \cdot \langle \mathbf{a}, \mathbf{b} \rangle$. The part $\mathbf{x} \otimes \mathbf{a}$ intuitively is encrypted by the EI-PLMCFE (*normal* encryption) during encryption of indexed EITT-MCFE, whereas the part $\mathbf{y} \otimes \mathbf{b}$ is extracted by the keygen of the EI-PLMCFE during key generation of indexed EITT-MCFE. The decryption involves dividing by $\langle \mathbf{a}, \mathbf{b} \rangle$, and this is used crucially in our tracing-encryption algorithm (subroutine of the main tracing, see Figure 4), for an index range n_{indx}: *(i)* the tracer switches to the *special encryption* algorithm of the underlying EI-PLMCFE scheme to teest index by index between $1, 2 \ldots, n_{\mathrm{indx}} + 1$, and *(ii)* the tracer send tracing signal such that running indexes in $[n_{\mathrm{indx}}]$, $\mathbf{a}$ is chosen as in the normal encryption but when the index tested is $n_{\mathrm{indx}} + 1$ the vector $\mathbf{a}$ makes $\langle \mathbf{a}, \mathbf{b} \rangle = 0$. This case of $n_{\mathrm{indx}} + 1$ allows choosing the tracing challenges to be $\mathbf{x}^{(0)} \neq \mathbf{x}^{(1)}$ such that $\langle \mathbf{x}^{(0)}, \mathbf{y} \rangle \neq \langle \mathbf{x}^{(1)}, \mathbf{y} \rangle$, for a good decoder to trivially distinguish, but from the view of the special encryption for $n_{\mathrm{indx}} + 1$ it still holds $\langle \mathbf{x}^{(0)} \otimes \mathbf{a}, \mathbf{y} \otimes \mathbf{b} \rangle = 0 = \langle \mathbf{x}^{(1)} \otimes \mathbf{a}, \mathbf{y} \otimes \mathbf{b} \rangle$. In the end, this implies a significant gap in the distinguishing probability of the decoder between $[n_{\mathrm{indx}}]$ and $n_{\mathrm{indx}} + 1$, helping the tracer of EITT-MCFE to identify the index j that creates the significant jump in the distinguishing probability $(j-1, j)$. Last but not least, our transformation in Section 4.2.1 generalizes this idea into the context of *multi-client* setting, while resolving a new challenge that the $\mathbf{x}$-vectors are now masked *independently and non-interactively* by each client i with a secret share of 0. The idea to achieve this masking resembles the MCFE construction in **Step 1** in Section 1.2.2.

4.2.1 Construction

Building blocks. We assume the existence of

- A secure n-client $\mathsf{EIPLMCFE}$ scheme for the inner product function $F_1 \colon \mathbb{Z}_q^{n^2 \cdot m \cdot k} \times \mathbb{Z}_q^{n^2 \cdot m \cdot k} \to \mathbb{Z}_q$ and identity space $\{\mathcal{ID} := \{0,1\}^\kappa\}_{\kappa \in \mathbb{N}}$. (So each client encrypts a vector of size $n \cdot m \cdot k$.)
- Two families of pseudorandom functions $\mathsf{PRF}(\mathsf{k}, \cdot), \mathsf{PRF}_2(\mathsf{k}, \cdot)$ of appropriate input/output length.

We will build a traceable MCFE scheme for the inner product function $F \colon \mathbb{Z}_q^{n \cdot m} \times \mathbb{Z}_q^{n \cdot m} \to \mathbb{Z}_q$ with identical identity space.

Notations. We will denote

- $\mathbf{X} = \mathbf{x}_1 \| \cdots \| \mathbf{x}_n \in \mathbb{Z}_q^{n \cdot m}$, where each $\mathbf{x}_i \in \mathbb{Z}_q^m$, the message (which will be encrypted by n clients).
- $\mathbf{Y} = \mathbf{y}_1 \| \cdots \| \mathbf{y}_n \in \mathbb{Z}_q^{n \cdot m}$, where each $\mathbf{y}_i \in \mathbb{Z}_q^m$, the function (which will be input to the KeyGen oracle).

Construction.

$\underline{\mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}})}$**:** Do the following:

- For $i \in [n], \forall j \neq i : \mathsf{k}_{i,j} = \mathsf{k}_{j,i} \xleftarrow{\$} \{0,1\}^\lambda$.
- Sample $\mathsf{k}_{\mathrm{ind}} \xleftarrow{\$} \{0,1\}^\lambda$.
- Run $\mathsf{EIPLMCFE.Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{indx}}) \to (\mathsf{msk\text{-}eipl}, (\mathsf{ek\text{-}eipl}_i)_{i \in [n]})$.
- We return $\mathsf{msk} \coloneqq (\mathsf{msk\text{-}eipl}, \{\mathsf{k}_{i,j}\}_{j>i}, \mathsf{k}_{\mathrm{ind}})$ and the encryption keys $\mathsf{ek}_i \coloneqq (\mathsf{ek\text{-}eipl}_i, \{\mathsf{k}_{i,j}\}_{j>i})$.

$\underline{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j, \mathbf{Y})}$**:** Parse $\mathsf{msk} \coloneqq (\mathsf{msk\text{-}eipl}, \{\mathsf{k}_{i,j}\}_{j>i}, \mathsf{k}_{\mathrm{ind}})$ and do the following:

- Compute $\mathbf{b} = \mathbf{b}_1 \| \cdots \| \mathbf{b}_n \leftarrow \mathsf{PRF}_2(\mathsf{k}_{\mathrm{ind}}, j) \in \mathbb{Z}_q^{n \cdot k}$.
- Let $\mathbf{z}_i = 0 \| \cdots \| \underbrace{\mathbf{Y} \otimes \mathbf{b}_i}_{i\text{-th coord. among } n} \| \cdots \| 0 \in \mathbb{Z}_q^{n^2 \cdot m \cdot k}$ for all $i \in [n]$.
- Output $\left((\mathsf{EIPLMCFE.KeyGen}(\mathsf{msk\text{-}eipl}, \mathsf{id}, j, \mathbf{z}_i))_{i \in [n]}, \mathbf{b} \right)$ as the decryption key.

$\underline{\mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)}$**:** Parse $\mathsf{ek}_i \coloneqq (\mathsf{ek\text{-}eipl}_i, \{\mathsf{k}_{i,j}\}_{j>i})$ and do the following:

- Compute $\mathbf{t}_{i,\mathsf{tag}} \leftarrow \sum_{j \neq i} (-1)^{j<i} \mathsf{PRF}(\mathsf{k}_{i,j}, \mathsf{tag}) \in \mathbb{Z}_q^{n \cdot m}$, where $(-1)^{j<i} \coloneqq -1$ if and only if $j < i$.
- Choose a random $\mathbf{a}_i \in \mathbb{Z}_q^k$.
- We define $\widetilde{\mathbf{w}}_i \coloneqq 0 \| \cdots \| 0 \| \mathbf{x}_i \| 0 \| \cdots \| 0 + \mathbf{t}_{i,\mathsf{tag}} \in \mathbb{Z}_q^{n \cdot m}$ and compute $\mathbf{w}_i = \widetilde{\mathbf{w}}_i \otimes \mathbf{a}_i \in \mathbb{Z}_q^{n \cdot m \cdot k}$.
- Return $(\mathsf{EIPLMCFE.Enc}(\mathsf{pp}, \mathsf{ek\text{-}eipl}_i, \mathsf{tag}, \mathbf{w}_i), \mathbf{a}_i)$.

$\underline{\mathsf{Dec}(\mathsf{pp}, \mathsf{dk}, \mathbf{c}_{\mathsf{tag}})}$**:** Do the following:

- Parse $\mathsf{dk} \coloneqq \left((\mathsf{dk\text{-}eipl}_i)_{i \in [n]}, \mathbf{b}_1 \| \cdots \| \mathbf{b}_n \right)$ and $\mathbf{c}_{\mathsf{tag}} \coloneqq \left(\mathsf{ct\text{-}eipl}_{i,\mathsf{tag}}, \mathbf{a}_i \right)_{i \in [n]}$.
- For each $i \in [n]$, we use the specific i-th component key $\mathsf{dk\text{-}eipl}_i$ to decrypt the total ensemble of EI-PLMCFE ciphertexts $(\mathsf{ct\text{-}eipl}_{i,\mathsf{tag}})_{i \in [n]}$ and obtain

$$\begin{aligned} e_i &= \frac{\mathsf{EIPLMCFE.Dec}(\mathsf{pp}, \mathsf{dk\text{-}eipl}_i, (\mathsf{ct\text{-}eipl}_{i,\mathsf{tag}})_{i \in [n]})}{\langle \mathbf{a}_i, \mathbf{b}_i \rangle} \\ &= \frac{\langle \widetilde{\mathbf{w}}_1 \otimes \mathbf{a}_1 \| \cdots \| \widetilde{\mathbf{w}}_n \otimes \mathbf{a}_n, 0 \| \cdots \| \mathbf{Y} \otimes \mathbf{b}_i \| \cdots \| 0 \rangle}{\langle \mathbf{a}_i, \mathbf{b}_i \rangle} \\ &= \frac{\langle \widetilde{\mathbf{w}}_i \otimes \mathbf{a}_i, \mathbf{Y} \otimes \mathbf{b}_i \rangle}{\langle \mathbf{a}_i, \mathbf{b}_i \rangle} \\ &= \frac{\langle \widetilde{\mathbf{w}}_i, \mathbf{Y} \rangle \cdot \langle \mathbf{a}_i, \mathbf{b}_i \rangle}{\langle \mathbf{a}_i, \mathbf{b}_i \rangle} \\ &= \langle \mathbf{x}_i, \mathbf{y}_i \rangle + \langle \mathbf{t}_{i,\mathsf{tag}}, \mathbf{Y} \rangle. \end{aligned}$$

- Return $\sum_i e_i$.

$\underline{\mathsf{Trace}^D(\mathsf{msk}, 1^{1/\epsilon}, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})}$**:** Let $y \coloneqq 1/\epsilon$. Do the following:

- Set $T^{\mathrm{index}} \coloneqq \emptyset$. For $j = 1$ to n_{indx}:
  - Compute $b, p, q \leftarrow \mathsf{IndexTrace}(\mathsf{msk}, 1^y, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, j)$, where the algorithm $\mathsf{IndexTrace}$ is given in Fig. 5.
  - If $b = 1$, set $T^{\mathrm{index}} = T^{\mathrm{index}} \cup \{(j, p, q)\}$.
- Set $T \coloneqq \emptyset$. For each $(j, p, q) \in T^{\mathrm{index}}$:
  - Compute $\mathsf{id} \leftarrow \mathsf{IdTrace}(\mathsf{msk}, 1^y, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, (j, p, q))$, where the algorithm $\mathsf{IdTrace}$ is given in Fig. 6.
  - Set $T = T \cup \{\mathsf{id}\}$.
- Output T as the set of traitors.

Input: Master secret key msk, a tag tag, a triple index-position-bit (j, ℓ, b), and messages $\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}$, and a challenge bit γ.
Algorithm: This algorithm will be used later in the tracing algorithm Trace. It does the following:

- Parse msk to get k_{ind}. Then for each $j' \in [n_{\text{indx}} + 1]$, re-compute

$$\mathbf{b}_{j'} := \mathbf{b}_{j',1} \| \cdots \| \mathbf{b}_{j',n} = \mathsf{PRF}_2(\mathsf{k}_{\text{ind}}, j') \in \mathbb{Z}_q^{n \cdot k},$$

  as in the first step of KeyGen.
- Parse msk to get PRF keys and re-compute $\mathbf{t}_{i,\mathsf{tag}} \in \mathbb{Z}_q^{n \cdot m}$ as in the first step of Enc.[a]
- Choose a random non-zero vector $\mathbf{v}_i \leftarrow\!\$\, \mathbb{Z}_q^k \setminus \{0\}$.
- Compute $\mathbf{v}_i^{(\gamma)}$ for $\gamma \in \{0, 1\}$ as:
  - If $j < n_{\text{indx}} + 1$, set $\mathbf{v}_i^{(0)} = \mathbf{v}_i^{(1)} = \mathbf{v}_i$.
  - If $j = n_{\text{indx}} + 1$, choose random $\mathbf{v}_i^{(0)} \neq \mathbf{v}_i^{(1)} \in \mathbb{Z}_q^k$ such that:
    1. $\langle \mathbf{b}_{j',i}, \mathbf{v}_i^{(0)} \rangle = 0$ for all $j' \in [n_{\text{indx}}]$.
    2. $\langle \mathbf{b}_{j',i}, \mathbf{v}_i^{(1)} \rangle = 0$ for all $j' \in [n_{\text{indx}}]$.
- Compute $\mathbf{w}_i^{(\gamma)} = (0\| \cdots \|\mathbf{x}_i^{(\gamma)}\| \cdots \|0 + \mathbf{t}_{i,\mathsf{tag}}) \otimes \mathbf{v}_i^{(\gamma)} \in \mathbb{Z}_q^{m \cdot k}$.
- Return
  - $(\mathsf{EIPLMCFE.SplEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, b), \mathbf{w}_i^{(\gamma)}), \mathbf{v}_i)$ if $j < n_{\text{indx}} + 1$.
  - $(\mathsf{EIPLMCFE.SplEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, b), \mathbf{w}_i^{(\gamma)}), \mathbf{v}_i^{(\gamma)})$, otherwise.

[a] Recall that i is the challenge index embedded in the messages.

Fig. 4. Algorithm $\mathsf{TraceEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, b), \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, \gamma)$.

Correctness. Correctness comes from the fact that for any tag, $\sum_i \mathbf{t}_{i,\mathsf{tag}} = 0$. The parameter k is chosen by Setup so that $2n_{\text{indx}} < k$, to guarantee the tracing algorithm works, i.e., the vectors $\mathbf{v}_i^{(0)}$ and $\mathbf{v}_i^{(1)}$ in Equations (1)-(2) are guaranteed to exist.

Theorem 2. *If* $\mathsf{EIPLMCFE}$ *is a 1-query secure EIPLMCFE as per Definition 18 to Definition 22, then construction in Section 4.2.1 is secure, as per*

- adaptive one-challenge *IND-CPA security with* complete *queries and* repetitions*, under* all-but-two adaptive corruption *(Definition 5),*
- *tracing security (Definition 16),*

where the admissibility condition is checked for the inner product function class (Definition 11) and all-but-two adaptive corruption *means at most* $(n-2)$*-among-n clients can be adaptively corrupted.*[17]

The proof of this theorem follows from the two following theorems.

Theorem 3. *If* $\mathsf{EIPLMCFE}$ *is a 1-query secure EIPLMCFE as per Definition 18 to Definition 22, then the construction in Section 4.2.1 is* adaptive one-challenge *IND-CPA security with* complete *queries and* repetitions*, under* all-but-two adaptive corruption *(Definition 5), where* all-but-two adaptive corruption *means at most* $(n-2)$*-among-n clients can be adaptively corrupted.*

Theorem 4. *If* $\mathsf{EIPLMCFE}$ *is a 1-query secure EIPLMCFE as per Definition 18 to Definition 22, then the construction in Section 4.2.1 is securely traceable (Definition 16).*

[17] We make this corruption restriction to ease the proof presentation. Similar techniques from [ABG19] can be used to treat the particular case of only 1 honest client.

Input: Master secret key msk, parameter y, a tag tag, two messages $\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}$ and an index j.
Algorithm:
Let $\epsilon = \lfloor 1/y \rfloor$. It sets $N = \lambda \cdot n_{\mathrm{indx}}/\epsilon$ and $\mathsf{count}_1 = \mathsf{count}_2 = 0$. For $\gamma = 1$ to N, it computes the following:

1. Choose $b_\gamma \leftarrow\!\$\ \{0,1\}$ and compute
$$\mathsf{ct}_{j,1} \leftarrow \mathsf{TraceEnc}(\mathsf{msk}, \mathsf{tag}, (j, \perp, 0), \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, b_\gamma)$$
and send $\mathsf{ct}_{j,1}$ to D, where $\mathsf{TraceEnc}$ is defined in Fig. 4. If D output b_γ, set $\mathsf{count}_1 = \mathsf{count}_1 + 1$, otherwise set $\mathsf{count}_1 = \mathsf{count}_1 - 1$.
2. Chose $c_\gamma \leftarrow\!\$\ \{0,1\}$ and compute
$$\mathsf{ct}_{j,2} \leftarrow \mathsf{TraceEnc}(\mathsf{msk}, \mathsf{tag}, (j+1, \perp, 0), \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, c_\gamma)$$
and send $\mathsf{ct}_{j,2}$ to D. If D output c_γ, set $\mathsf{count}_2 = \mathsf{count}_2 + 1$, otherwise set $\mathsf{count}_2 = \mathsf{count}_2 - 1$.

If $\frac{\mathsf{count}_1 - \mathsf{count}_2}{N} > \frac{\epsilon}{4n_{\mathrm{indx}}}$, output $(1, \frac{\mathsf{count}_1}{N}, \frac{\mathsf{count}_2}{N})$, else output $(0, \perp, \perp)$.

Fig. 5. Algorithm $\mathsf{IndexTrace}(\mathsf{msk}, 1^y, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, j)$.

Input: Master secret key msk, parameter y, a tag tag, two messages $\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}$ and an index j and probabilities p, q.
Algorithm:
Let $\epsilon = \lfloor 1/y \rfloor$. It sets $N = \lambda \cdot n_{\mathrm{indx}}/\epsilon$ and $\mathsf{count}_\ell = 0$ for $\ell \in [\kappa]$.

1. For each $\ell = 1$ to κ, for $\gamma = 1$ to N, choose $b_\gamma \leftarrow\!\$\ \{0,1\}$ and compute
$$\mathsf{ct}_j \leftarrow \mathsf{TraceEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, 0), \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, b_\gamma)$$
and send ct_j to D. If D output b_γ, set $\mathsf{count}_\ell = \mathsf{count}_\ell + 1$, otherwise set $\mathsf{count}_\ell = \mathsf{count}_\ell - 1$.
2. It then sets $\mathsf{id} \coloneqq \perp$. For $\ell = 1$ to κ, it computes the following: if $\frac{p+q}{2} > \frac{\mathsf{count}_\ell}{N}$, set $\mathsf{id}_\ell = 0$, else set $\mathsf{id}_\ell = 1$.
3. Finally, output id.

Fig. 6. Algorithm $\mathsf{IdTrace}(\mathsf{msk}, 1^y, \mathsf{tag}, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, (j, p, q))$.

The proof of Theorem 3 is given in Section 4.2.1, and the proof of Theorem 4 is given in Section 4.2.1.

4.2.2 Proof of Theorem 3

We translate the *strong* admissibility from Definition 4 for general function classes to the case of inner products. These concrete conditions are used in the proof of Theorem 3.

Proof of Theorem 3. We give the main ideas of the game transitions. The changes that make the transitions between games are highlighted in gray. The advantage of an adversary $\mathcal{A}$ in a game G_i is denoted by

$$\mathsf{Adv}(\mathsf{G}_i) \coloneqq \Pr[\mathsf{G}_i = 1] \ .$$

G_0**:** This is the original security game following Definition 5, where the challenge bit is $b = 0$. We recall that we impose the one challenge restriction, where the only challenge tag in $(\star, \star, \mathsf{tag})$ to **LoR** can be repetitively queried to both **Enc** and **LoR** with different $\mathbf{x}_i$ for the same i. For simplicity, we add a constraint that the challenge tag tag is *not* queried to **Enc**. This incurs a multiplicative loss factor in advantage up to an inverse of polynomial in λ, where we can reduce to the normal one challenge restriction by guessing the challenge tag among the tags for encryption, and responding all of its **Enc** queries $(i, \mathbf{x}_i, \mathsf{tag})$ by $\mathbf{LoR}(i, \mathbf{x}_i, \mathbf{x}_i, \mathsf{tag})$.

G_1**:** The simulator guesses the number of honest clients $h \leq n$ by sampling $h \xleftarrow{\$} [n]$, *i.e.* equivalently at the end of the game $|\mathcal{H}| = h$ and $|\mathcal{C}| = n - h$ is the number of *adaptively* corrupted clients. If the guess h is incorrect before **Finalize**, the simulator aborts and outputs 0. The correct guess happens with probability $1/(n+1)$ and we can calculate that $\mathsf{Adv}(\mathsf{G}_1) \leq 1/(n+1) \cdot \mathsf{Adv}(\mathsf{G}_0)$.

G_2**:** The set of honest clients is ordered by appearance and indexed by $\mathcal{H} = \{i_1, \ldots, i_h\}$, where $h \geq 2$ results from the guess of the size of $\mathcal{H}$ from the previous game. The simulator samples $\mathbf{u}_2, \ldots, \mathbf{u}_h \xleftarrow{\$} \mathbb{Z}_q^{n \cdot m}$ and computes $\mathbf{t}_{\tilde{i},\mathsf{tag}}$ for each $i \in \mathcal{H}$ below, changes are grayed:

- **If** $\tilde{i} = i_1$, define

$$\mathbf{t}_{\tilde{i},\mathsf{tag}} := \sum_{j \neq \tilde{i}} (-1)^{j<\tilde{i}} \mathsf{PRF}(\mathsf{k}_{\tilde{i},j}, \mathsf{tag}) + \boxed{\textstyle\sum_{\ell=2}^{h} \mathbf{u}_\ell} \in \mathbb{Z}_q^{n \cdot m} \ .$$

- **If** $\tilde{i} = i_\ell$ for $\ell \in \{2, 3, \ldots, h\}$, define

$$\mathbf{t}_{\tilde{i},\mathsf{tag}} := \sum_{j \neq \tilde{i}} (-1)^{j<\tilde{i}} \mathsf{PRF}(\mathsf{k}_{\tilde{i},j}, \mathsf{tag}) - \boxed{\mathbf{u}_\ell} \in \mathbb{Z}_q^{n \cdot m} \ .$$

The rest of the game is unchanged, where the simulation of **Setup**, **Extract**, and **LoR** queries is identical to the real game using the set up parameters. We remark that the order of apperance of an honest $\tilde{i}$ is defined *as per* the queries to **LoR**, which involves the above calculation of $\mathbf{t}_{\tilde{i},\mathsf{tag}}$.

We argue the indistinguishability between G_1 and G_2 by a hybrid argument, following a sequence $\mathsf{G}_1 = \mathsf{G}_{1.0}, \mathsf{G}_{1.1}, \ldots, \mathsf{G}_{1.h} = \mathsf{G}_2$ in which the calculation of $\mathbf{t}_{i,\mathsf{tag}}$ is done in $\mathsf{G}_{1.\mu}$ for $\mu \in [h]$ below, highlighted in boxes:

- **If** $\mu = 1$, $\mathsf{G}_{1.\mu}$ is identical to $\mathsf{G}_{1.\mu-1}$.
- **If** $\mu \geq 2$, for $\tilde{i} = i_1$, define

$$\mathbf{t}_{\tilde{i},\mathsf{tag}} := \sum_{j \neq \tilde{i}} (-1)^{j<\tilde{i}} \mathsf{PRF}(\mathsf{k}_{\tilde{i},j}, \mathsf{tag}) + \boxed{\textstyle\sum_{\ell=2}^{\mu} \mathbf{u}_\ell} \in \mathbb{Z}_q^{n \cdot m} \ .$$

- **If** $\mu \geq 2$, for $\tilde{i} = i_\ell$ for $\ell \in \{2, 3, \ldots, \mu\}$, define

$$\mathbf{t}_{\tilde{i},\mathsf{tag}} := \sum_{j \neq \tilde{i}} (-1)^{j<\tilde{i}} \mathsf{PRF}(\mathsf{k}_{\tilde{i},j}, \mathsf{tag}) - \boxed{\mathbf{u}_\ell} \in \mathbb{Z}_q^{n \cdot m} \ .$$

- **If** $\mu \geq 2$, for $\tilde{i} = i_\ell$ for $\ell \in \{\mu + 1, \ldots, h\}$,

$$\boxed{\mathbf{t}_{\tilde{i},\mathsf{tag}} := \textstyle\sum_{j \neq \tilde{i}} (-1)^{j<\tilde{i}} \mathsf{PRF}(\mathsf{k}_{\tilde{i},j}, \mathsf{tag}) \in \mathbb{Z}_q^{n \cdot m}} \ .$$

Roughly speaking, in the game $\mathsf{G}_{1.\mu}$ where $2 \leq \mu \leq h$, we embed a μ-out-of-μ secret sharing of 0 among the first μ honest clients, ordered by appearance in $\mathcal{H} = \{i_1, \ldots, i_h\}$. The hybrid argument consists of two steps: *(i)* Going from $\mathsf{G}_{1.1}$ to $\mathsf{G}_{1.2}$, and *(ii)* Going from $\mathsf{G}_{1.\mu}$ to $\mathsf{G}_{1.\mu+1}$ for $\mu \in \{2, 3, \ldots, h\}$. We give details below:

- **Step (i):** The transition from $\mathsf{G}_{1.1}$ to $\mathsf{G}_{1.2}$ is indistinguishable by the PRF security. The reduction first guesses the first two honest i_1, i_2 and samples $\mathbf{u}_2 \xleftarrow{\$} \mathbb{Z}_q^{n \cdot m}$. If the guess is incorrect, the reduction

aborts and outputs 0. Otherwise, for i_1, the reduction computes

$$\mathbf{t}_{i_1,\mathsf{tag}} = \sum_{j \notin \{i_1,i_2\}} (-1)^{j<i_1} \mathsf{PRF}(\mathsf{k}_{i_1,j}, \mathsf{tag}) + \mathsf{ChPRF}((i_1,i_2), \mathsf{tag}) + \mathbf{u}_2$$

whereas for i_2, the reduction computes

$$\mathbf{t}_{i_2,\mathsf{tag}} = \sum_{j \notin \{i_1,i_2\}} (-1)^{j<i_2} \mathsf{PRF}(\mathsf{k}_{i_2,j}, \mathsf{tag}) - \mathsf{ChPRF}((i_1,i_2), \mathsf{tag}) - \mathbf{u}_2$$

where $\mathsf{ChPRF}((i_1,i_2),\mathsf{tag})$ is the challenge PRF query with respect to the key k_{i_1,i_2}. We remark that both i_1, i_2 are honest and the PRF key k_{i_1,i_2} is never revealed to the adversary by construction of ek_{i_1} containing $\{\mathsf{k}_{i_1,i_2} = \mathsf{k}_{i_2,i_1}\}$ and the latter is in ek_{i_2}. We apply the PRF security twice:

$$\underbrace{\mathsf{PRF}(\mathsf{k}_{i_1,i_2}, \mathsf{tag})}_{\text{in } \mathsf{G}_{1.1}} \overset{(*)}{\sim_c} \mathsf{RF}_{(i_1,i_2)}(\mathsf{tag}) \overset{(\dagger)}{\equiv} \mathsf{RF}_{(i_1,i_2)}(\mathsf{tag}) + \mathbf{u}_2 \overset{(**)}{\sim_c} \overbrace{\mathsf{PRF}(\mathsf{k}_{i_1,i_2}, \mathsf{tag}) + \mathbf{u}_2}^{\text{in } \mathsf{G}_{1.2}} \ ,$$

where $(*)$ follows from the PRF security, $(\dagger)$ follows from the fact that $\mathbf{u}_2$ is uniformly random, and $(**)$ follows from the PRF security. The transition is indistinguishable by the PRF security and the correct guess probability of $\frac{2}{n(n-1)}$ for guess i_1, i_2.

- **Step (ii):** The transition from $\mathsf{G}_{1.\mu}$ to $\mathsf{G}_{1.\mu+1}$ for $\mu \in \{2, 3, \ldots, h-1\}$ is indistinguishable by the PRF security. The reduction first guesses the next honest client $i_{\mu+1}$ and samples $\mathbf{u}_{\mu+1} \xleftarrow{\$} \mathbb{Z}_q^{n \cdot m}$. The reduction performs similarly as in the case of i_1, i_2 in Step (i), by querying the PRF challenge with respect to the key $k_{i_\mu,i_{\mu+1}}$. It is important to recall that $i_\mu, i_{\mu+1}$ are honest and the PRF key $k_{i_\mu,i_{\mu+1}}$ is never revealed to the adversary by construction of ek_{i_μ} containing $\{\mathsf{k}_{i_\mu,i_{\mu+1}} = \mathsf{k}_{i_{\mu+1},i_\mu}\}$ and the latter is in $\mathsf{ek}_{i_{\mu+1}}$. Finally, is indistinguishable by the PRF security and the correct guess probability of $\frac{1}{n}$ for guess $i_{\mu+1}$.

A sequence of the above steps results in the indistinguishability between $\mathsf{G}_1 = \mathsf{G}_{1.0}$ and $\mathsf{G}_2 = \mathsf{G}_{1.h}$.

G_3: We switch the vector $\mathbf{b}_{\mathsf{id},j}$ into a uniformly random vector

$$\mathbf{b}_{\mathsf{id},j,1} \| \cdots \| \mathbf{b}_{\mathsf{id},j,n} \xleftarrow{\$} \mathbb{Z}_q^{n \cdot k} \ ,$$

noting that the PRF key $\mathsf{k}_{\mathrm{ind}}$ is part of the master secret key msk and never revealed to the adversary. We index the vector $\mathbf{b}_{\mathsf{id},j}$ by id, j that is associated to each **Extract** query. The transition is indistinguishable under the PRF security.

G_4: For $i \in \mathcal{H}$ honest that are ordered by appearance and resulting from the games G_1 and G_2, the corresponding challenge ciphertext component is computed differently where we define

$$\widetilde{\mathbf{w}}_i \coloneqq 0 \| \cdots \| 0 \| \mathbf{x}_i^{(1)} \| 0 \| \cdots \| 0 + \mathbf{t}_{i,\mathsf{tag}} \in \mathbb{Z}_q^{n \cdot m} \ ,$$

and compute $\mathbf{w}_i = \widetilde{\mathbf{w}}_i \otimes \mathbf{a}_i \in \mathbb{Z}_q^{n \cdot m \cdot k}$. We perform a reduction to the underlying security of the EI-PLMCFE scheme below:

1. Our simulator starts the IND-CPA game with the challgenger against the EI-PLMCFE scheme. The **Initialize** is run so that the challenger generates $(\mathsf{msk\text{-}eipl}, (\mathsf{ek\text{-}eipl}_i)_{i \in [n]})$ for the EI-PLMCFE. The simulator sets up the rest of the msk by sampling for $i \in [n], \forall j \neq i : \mathsf{k}_{i,j} = \mathsf{k}_{j,i} \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{k}_{\mathrm{ind}} \xleftarrow{\$} \{0,1\}^\lambda$, as in the previous game. When the adversary $\mathcal{A}$ outputs a static corruption set $\mathcal{C}$, the simulator demands the challenger to corrupt the corresponding $i \in \mathcal{C}$ and returns to $\mathcal{A}$ for every $i \in \mathcal{C}$ the key $\mathsf{ek}_i \coloneqq (\mathsf{ek\text{-}eipl}_i, \{\mathsf{k}_{i,j}\}_{j>i})$.

2. Upon an **Extraction** query for $\mathsf{id}, j, \mathbf{Y}$ by the adversary $\mathcal{A}$, the simulator samples $\mathbf{b}_{\mathsf{id},j} = \mathbf{b}_{\mathsf{id},j,1} \| \dots \| \mathbf{b}_{\mathsf{id},j,n} \xleftarrow{\$} \mathbb{Z}_q^{n \cdot k}$ as in G_3, then $\mathbf{z}_i = 0 \| \cdots \| \underbrace{\mathbf{Y} \otimes \mathbf{b}_i}_{i\text{-th coord. among } n} \| \cdots \| 0 \in \mathbb{Z}_q^{n^2 \cdot m \cdot k}$ for all $i \in [n]$. For each $i \in [n]$ our simulator queries to the EI-PLMCFE-challenger on $(\mathsf{msk\text{-}eipl}, \mathsf{id}, j, \mathbf{z}_i)$ and obtain $\mathsf{dk}_i^{\mathsf{eipl}}$. Finally, the simulator returns $\left((\mathsf{dk}_i^{\mathsf{eipl}})_{i \in [n]}, \mathbf{b}_{\mathsf{id},j}\right)$ to the adversary.

3. Upon a **LoR** query for $(i, \mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}, \mathsf{tag})$ by the adversary $\mathcal{A}$, the simulator:

   - samples $\mathbf{r}_{i,\mathsf{tag}} \xleftarrow{\$} \mathbb{Z}_q^{n \cdot m}$ for $i \in \mathcal{H}$ so that they form a sharing of $\mathbf{0}$, and sets the share $\mathbf{t}_{i,\mathsf{tag}} \leftarrow \mathbf{r}_{i,\mathsf{tag}}$, as in G_2
   - samples $b \xleftarrow{\$} \{0,1\}$, $\mathbf{a}_i \xleftarrow{\$} \mathbb{Z}_q^k$ as in the game for $\mathcal{A}$, then computes

   $$\widetilde{\mathbf{w}}_i^{(\mathsf{eipl},0)} := 0 \| \cdots \| 0 \| \mathbf{x}_i^{(b)} \| 0 \| \cdots \| 0 + \mathbf{t}_{i,\mathsf{tag}} \in \mathbb{Z}_q^{n \cdot m} \ ,$$
   $$\widetilde{\mathbf{w}}_i^{(\mathsf{eipl},1)} := 0 \| \cdots \| 0 \| \mathbf{x}_i^{(1)} \| 0 \| \cdots \| 0 + \mathbf{t}_{i,\mathsf{tag}} \in \mathbb{Z}_q^{n \cdot m}$$

   and computes $\mathbf{w}_i^{(\mathsf{eipl},0)} = \widetilde{\mathbf{w}}_i^{(\mathsf{eipl},0)} \otimes \mathbf{a}_i$ and $\mathbf{w}_i^{(\mathsf{eipl},1)} = \widetilde{\mathbf{w}}_i^{(\mathsf{eipl},1)} \otimes \mathbf{a}_i$
   - sends an **LoR** query $(i, \mathbf{w}_i^{(\mathsf{eipl},0)}, \mathbf{w}_i^{(\mathsf{eipl},1)}, \mathsf{tag})$ to the EI-PLMCFE-**LoR**, receives the ciphertext $\mathsf{ct}_i^{(\mathsf{eipl},b')}$, in which $b' \xleftarrow{\$} \{0,1\}$ is the EI-PLMCFE challenger's bit.
   - returns $(\mathsf{ct}_i^{(\mathsf{eipl},b')}, \mathbf{a}_i)$ to the adversary $\mathcal{A}$.

   The above simulation is done for each **LoR** query by $\mathcal{A}$, with respect to repetitions j_i for each $i \in [n]$. Thanks to the fresh randomness $\mathbf{a}_i^{j_i} \xleftarrow{\$} \mathbb{Z}_q^k$ at step 3 for a repetition j_i, each repetitive **LoR** $(i, \mathbf{x}_i^{(0,j_i)}, \mathbf{x}_i^{(1,j_i)}, \mathsf{tag})$ query by $\mathcal{A}$, indexed by j_i for fixed $i \in [n]$ and challenge tag tag, will induce a different $(i, \mathbf{w}_i^{(\mathsf{eipl},0,j_i)}, \mathbf{w}_i^{(\mathsf{eipl},1,j_i)}, \mathsf{tag})$ query to the EI-PLMCFE-**LoR** at repetition j_i for the same (i, tag)

4. Upon an **Enc** query for $(i, \mathbf{x}_i, \mathsf{tag})$ by the adversary $\mathcal{A}$, the simulator performs a similar simulation as in the previous step 3, except that the simulator sends an **Enc** query $(i, \mathbf{w}_i^{(\mathsf{eipl})}, \mathsf{tag}')$ to the EI-PLMCFE-**Enc** that encrypts the message $\mathbf{w}_i^{(\mathsf{eipl})}$ under the tag tag':

   $$\widetilde{\mathbf{w}}_i^{(\mathsf{eipl})} := 0 \| \cdots \| 0 \| \mathbf{x}_i \| 0 \| \cdots \| 0 + \mathbf{t}_{i,\mathsf{tag}'} \in \mathbb{Z}_q^{n \cdot m} \ ,$$

   where the share $\mathbf{t}_{i,\mathsf{tag}'}$ is computed as in G_2. As previously, the simulation is done with respect to each repetition j_i for each $i \in [n]$. We recall that thanks to G_0, the challenge tag tag is not queried to the EI-PLMCFE-**Enc**.

We argue that the adversary's view in G_4 is computationally indistinguishable from the real game G_3:

- When the EI-PLMCFE-**LoR** is queried with $(i, \mathbf{w}_i^{(0)}, \mathbf{w}_i^{(1)}, \mathsf{tag})$, when $b' = 0$, the returned ciphertext $\mathsf{ct}_i^{(\mathsf{eipl},0)}$ is computed as in G_3, with the challenge $\mathbf{x}_i^{(b)}$. An i-slot decryption vis-à-vis some functional decryption key $\left((\mathsf{dk}_i^{\mathsf{eipl}})_{i \in [n]}, \mathbf{b}_{\mathsf{id},j}\right)$ returns

$$e_i = \frac{\mathsf{EIPLMCFE.Dec}(\mathsf{pp}, \mathsf{dk}_i^{\mathsf{eipl}}, (\mathsf{ct}_i^{(\mathsf{eipl},1)})_{i \in [n]})}{\langle \mathbf{a}_i, \mathbf{b}_i \rangle} = \langle \mathbf{x}_i^{(b)}, \mathbf{y}_i \rangle + \langle \mathbf{t}_{i,\mathsf{tag}}, \mathbf{Y} \rangle \ .$$

- When $b' = 1$, the returned ciphertext $\mathsf{ct}_i^{(\mathsf{eipl},1)}$ encrypts the challenge $\mathbf{x}_i^{(1)}$, as in G_4. Necessarily, we verify that an i-slot decryption vis-à-vis some functional decryption key $\left((\mathsf{dk}_i^{\mathsf{eipl}})_{i\in[n]}, \mathbf{b}_{\mathsf{id},j}\right)$ leads to

$$\begin{aligned} e_i &= \frac{\mathsf{EIPLMCFE.Dec}(\mathsf{pp}, \mathsf{dk}_i^{\mathsf{eipl}}, (\mathsf{ct}_i^{(\mathsf{eipl},1)})_{i\in[n]})}{\langle \mathbf{a}_i, \mathbf{b}_i\rangle} \\ &= \frac{\langle \widetilde{\mathbf{w}}_1^{(\mathsf{eipl},1)} \otimes \mathbf{a}_1 \| \cdots \| \widetilde{\mathbf{w}}_n^{(\mathsf{eipl},1)} \otimes \mathbf{a}_n, 0 \| \cdots \| \mathbf{Y} \otimes \mathbf{b}_i \| \cdots \| 0\rangle}{\langle \mathbf{a}_i, \mathbf{b}_i\rangle} \\ &= \frac{\langle \widetilde{\mathbf{w}}_i^{(\mathsf{eipl},1)} \otimes \mathbf{a}_i, \mathbf{Y} \otimes \mathbf{b}_i\rangle}{\langle \mathbf{a}_i, \mathbf{b}_i\rangle} \\ &= \langle \mathbf{x}_i^{(b)}, \mathbf{y}_i\rangle + \langle \mathbf{t}_{i,\mathsf{tag}}, \mathbf{Y}\rangle + \langle \Delta\mathbf{x}_i, \mathbf{y}_i\rangle. \end{aligned}$$

  First of all, thanks to the setting of $\mathbf{t}_{i,\mathsf{tag}} := \mathbf{r}_{i,\mathsf{tag}} \xleftarrow{\$} \mathbb{Z}_q^{n\cdot m}$ from G_2, the term $\langle \mathbf{t}_{i,\mathsf{tag}}, \mathbf{Y}\rangle$ preserves the distribution of the partial term e_i, in comparison to the case $b' = 0$. We then use that fact that $\Delta\mathbf{x}_i = \mathbf{x}_i^{(1,j_i)} - \mathbf{x}_i^{(b,j_i)}$ is the difference of the challenge plaintext component, at repetition j_i for slot i, and is constant over the repetitions j_i thanks to condition 1 for the admissiblity with respect to *all* repetitions (note that for $i \in \mathcal{C}$ the added $\langle \Delta\mathbf{x}_i, \mathbf{y}_i\rangle = 0$ is trivial, by condition 2). Furthermore, once again, thanks to condition 1 it holds that $\sum_{i\in\mathcal{H}} \langle \Delta\mathbf{x}_i, \mathbf{y}_i\rangle = 0$ and thus the function evaluation is exactly preserved.
- As a consequence, if $\mathcal{A}$ is an admissibile adversary against the IND-CPA security game for the Indexed EITT, then our simulator is an admissible adversary against the IND-CPA security game for the underlying EI-PLMCFE scheme. The security of EI-PLMCFE under consideration is IND-CPA with *one-challenge*, *complete* ciphertexts, and *all-but-two adaptive corruption*, while being resilient against *adaptive* adversaries with *repetitions*.

  This concludes that the advantage of $\mathcal{A}$ in distinguishing G_3 and G_4 is upper bounded by the advantage of the simulator in winning the IND-CPA game against the underlying EI-PLMCFE.

In game G_4, the challenge bit b is not involved in the computation of the ciphertext, and the adversary's view is independent of b. This concludes the proof. □

4.2.3 Proof of Theorem 4

Our proof of correctness of tracing follows almost identically as that of [GKW19], with modifications in computing the correct trace probability, adapted to the multi-client setting. In particular, we show that the false trace probability is bounded by a negligible function, and the correct trace probability is close to the probability of $\mathcal{A}$ outputting an ϵ-successful decoding box for some non-negligible ϵ.

We first define some notations. Given any pirate decoder box D, any tag tag, and two messages $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, for any $j \in [n_{\mathrm{indx}} + 1], \ell \in [\kappa]$, let $\mathbf{w}_i^{(0)}, \mathbf{w}_i^{(1)}$ be the messages created from $\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)}$, respectively, together with some uniformly random vector $\mathbf{v}_i$ as in the Return step of $\mathsf{TraceEnc}$, where we use the index i to identify the i-th client's encryption slot. We define

$$\begin{aligned} p^D_{j,\perp} &:= \Pr\left[D\left((\mathsf{ct}_i, \mathbf{v}_i)_{i\in[n]}\right) = b \;\middle|\; \begin{array}{l} b \leftarrow\!\$\, \{0,1\} \\ \mathsf{ct}_i \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j, \perp, 0), \mathbf{w}_i^{(b)}) \end{array}\right] \\ p^D_{j,\ell} &:= \Pr\left[D\left((\mathsf{ct}_i, \mathbf{v}_i)_{i\in[n]}\right) = b \;\middle|\; \begin{array}{l} b \leftarrow\!\$\, \{0,1\} \\ \mathsf{ct}_i \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, (j, \ell, 0), \mathbf{w}_i^{(b)}) \end{array}\right] \\ p^D_{\mathrm{nrml}} &:= \Pr\left[D\left((\mathsf{ct}_i, \mathbf{v}_i)_{i\in[n]}\right) = b \mid b \leftarrow\!\$\, \{0,1\}, \mathsf{ct}_i \leftarrow \mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, \mathbf{w}_i^{(b)})\right] \end{aligned}$$

where the probability is taken over random coins of decoder D as well as the randomness used during encryption of $\mathsf{EIPLMCFE}$. For simplicity of notation, we will drop dependence on decoder D whenever it is clear from context.

In the following, we note that the generation of $\mathbf{w}_i^{(b)}$ for $b \in \{0,1\}$ in $\mathsf{TraceEnc}$ is exactly the same as that in Enc for all user-indices $j \in [n_{\mathrm{indx}}]$. The only difference is that $\mathsf{TraceEnc}$ uses the special encryption algorithm SplEnc instead of the normal encryption algorithm Enc. Looking ahead, this allows us to relate the probability p_{nrml} of the normal encryption in the tracing game to the probabilities $p_{j,\ell}$ of the special encryption in the tracing algorithm. For the exception in the case of the index $j = n_{\mathrm{indx}} + 1$, we give a formal treatment in the proof of Claim 4.1. We note that the same argument can be used anywhere involving indistinguishability on this index $j = n_{\mathrm{indx}} + 1$.

We show that false trace probability is upper bounded by some negligible function.

Lemma 1. *If the scheme* $\mathsf{EIPLMCFE}$ *is a 1-query secure EIPLMCFE scheme as per Definition 18 to Definition 22, then for every* PPT *adversary* $\mathcal{A}$*, polynomial* $q(\cdot)$ *and non-negligible function* $\epsilon(\cdot)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *satisfying* $\epsilon(\lambda) > 1/q(\lambda)$*,*

$$\texttt{Pr-FalseTr}_{\mathcal{A},\epsilon}(\lambda) \leq \mathsf{negl}(\lambda),$$

where $\texttt{Pr-FalseTr}_{\mathcal{A},\epsilon}(\cdot)$ *is defined in Definition 16.*

Proof. The proof of this lemma is similar to that in [GKW19, Theorem 5.2]. We sketch the high level idea of the proof and inform the reader modifications in their proof adapted to our multi-client setting.

Let $S \subseteq [n_{\mathrm{indx}}] \times \{0,1\}^\kappa$ be the set of index-identity pairs queried by the adversary $\mathcal{A}$ for decryption keys, $S_{\mathrm{indx}} \subseteq [n_{\mathrm{indx}}]$ be the set of indices queried by the adversary $\mathcal{A}$ for decryption keys, and let D be the decoder box output by $\mathcal{A}$.

In the following we skip the dependence of $\varepsilon(\cdot)$ on λ for simplicity of notation. For $j \in [n_{\mathrm{indx}}], \ell \in [\kappa]$, we define events

$$\begin{aligned}
\mathsf{Diff\text{-}Adv}_j^D &: p_{j,\perp}^D - p_{j+1,\perp}^D > \varepsilon/8n_{\mathrm{indx}} \\
\mathsf{Diff\text{-}Adv}_{j,\ell,\mathrm{lwr}}^D &: p_{j,\perp}^D - p_{j,\ell}^D > \varepsilon/16n_{\mathrm{indx}} \\
\mathsf{Diff\text{-}Adv}_{j,\ell,\mathrm{upr}}^D &: p_{j,\ell}^D - p_{j+1,\perp}^D > \varepsilon/16n_{\mathrm{indx}} \\
\mathsf{Diff\text{-}Adv}^D &: \bigvee_{j \in [n_{\mathrm{indx}}] \setminus S_{\mathrm{indx}}} \mathsf{Diff\text{-}Adv}_j^D \bigvee_{(j,\mathsf{id}) \in S, \ell \in [\kappa] \text{ s.t. } \mathsf{id}_\ell = 1} \mathsf{Diff\text{-}Adv}_{j,\ell,\mathrm{lwr}}^D \\
&\qquad \bigvee_{(j,\mathsf{id}) \in S, \ell \in [\kappa] \text{ s.t. } \mathsf{id}_\ell = 0} \mathsf{Diff\text{-}Adv}_{j,\ell,\mathrm{upr}}^D.
\end{aligned}$$

Next, note that the probability of the event *false trace* can be rewritten (using union bound) as follows by conditioning on the events defined above:

$$\begin{aligned}
\Pr[\mathsf{Fal\text{-}Tr}] \leq \Pr\left[\mathsf{Fal\text{-}Tr} \mid \overline{\mathsf{Diff\text{-}Adv}}\right] &+ \sum_{j \in [n_{\mathrm{indx}}]} \Pr[j \notin S_{\mathrm{indx}} \wedge \mathsf{Diff\text{-}Adv}_j] \\
&+ \sum_{(j,\ell) \in [n_{\mathrm{indx}}] \times [\kappa]} \Pr\left[\exists \mathsf{id} \in \{0,1\}^\kappa \text{ s.t. } (j,\mathsf{id}) \in S \wedge \begin{pmatrix} (\mathsf{Diff\text{-}Adv}_{j,\ell,\mathrm{lwr}} \wedge \mathsf{id}_\ell = 1) \\ \vee (\mathsf{Diff\text{-}Adv}_{j,\ell,\mathrm{upr}} \wedge \mathsf{id}_\ell = 0) \end{pmatrix}\right].
\end{aligned}$$

The first term is bounded by a negligible function identically as in [GKW19, Lemma 5.2]. The second term is bounded by a negligible function as proven in Lemma 2. The third one is bounded similarly as that of [GKW19, Lemma 5.4], whose proof is adapted to our setting as in Lemma 2. The lemma follows. □

Lemma 2. *If the scheme* $\mathsf{EIPLMCFE}$ *is a 1-query index hiding secure EI-PLMCFE scheme as per Definition 19, then for every* PPT *adversary* $\mathcal{A}$*, polynomial* $q(\cdot)$ *and non-negligible function* $\varepsilon(\cdot)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *satisfying* $\varepsilon(\lambda) > 1/q(\lambda)$ *and* $j \in [n_{\mathrm{indx}}]$*,*

$$\Pr[j \notin S_{\mathrm{indx}} \wedge \mathsf{Diff\text{-}Adv}_j] \leq \mathsf{negl}(\lambda).$$

Proof. Suppose, on the contrary, there exists a PPT adversary $\mathcal{A}$, polynomial $q(\cdot)$ and non-negligible functions $\varepsilon(\cdot), \delta(\cdot)$, such that for all $\lambda \in \mathbb{N}$ satisfying $\varepsilon(\lambda) > 1/q(\lambda)$, there exists an $j \in [n_{\text{indx}}]$ such that $\Pr[j \notin S_{\text{indx}} \wedge \mathsf{Diff\text{-}Adv}_j] \geq \delta(\lambda)$. Then we can use $\mathcal{A}$ to build a PPT reduction algorithm $\mathcal{B}$ that breaks the index hiding security property of EIPLMCFE.

The reduction algorithm $\mathcal{B}$ first receives $(1^{n_{\text{indx}}}, 1^{\kappa})$ from the adversary $\mathcal{A}$, which it forwards to the challenger. It then receives the EIPLMCFE public parameters pp from the challenger, which it sends to $\mathcal{A}$. Next, it randomly guesses the index hiding challenger with which it interacts by choosing an index $j^* \leftarrow [n_{\text{indx}}]$ and sends it to the EIPLMCFE challenger. The adversary $\mathcal{A}$ then queries to its oracles. $\mathcal{B}$ forwards $\mathcal{A}$'s queries on Enc and $\mathsf{Corrupt}$ oracles to its corresponding oracles. If $\mathcal{A}$ queries KeyGen on index j^*, $\mathcal{B}$ aborts and sends a random guess to the EIPLMCFE challenger. Else, on KeyGen query for (j, id) from $\mathcal{A}$, $\mathcal{B}$ forwards (j, id) to the EIPLMCFE challenger and forwards the challenger's response to the adversary $\mathcal{A}$. After all queries, the adversary sends a decoding box D, messages $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, and a tag tag to $\mathcal{B}$. The reduction algorithm then chooses two bits α, β uniformly at random. Next, $\mathcal{B}$ sends $(\mathsf{tag}, \mathbf{x}^{(\alpha)})$ to the EIPLMCFE challenger, and receives back a challenge ciphertext $\mathsf{ct}^{(b)}$. It also queries the EIPLMCFE challenger for a special-encryption of $\mathbf{x}^{(\alpha)}$ for index-position-value tuple $(j^* + \beta, \perp, 0)$. Let $\mathsf{ct}^{(\beta)}$ be the challenger's response. Finally, $\mathcal{B}$ runs decoder box D on $\mathsf{ct}^{(\beta)}$ and $\mathsf{ct}^{(b)}$ independently, and if $D(\mathsf{ct}^{(b)}) = D(\mathsf{ct}^{(\beta)})$, it outputs $b' = \beta$, else it outputs $b' = 1 - \beta$ as its guess.

First, note that $\mathcal{B}$ is an admissible adversary in the index hiding security game. This is because $\mathcal{B}$ does not query KeyGen on index-identity pair (j, id) such that $j = j^*$. Additionally, it only makes a single special-encryption query on index-position-value tuple $(j^*, \perp, 0)$ or $(j^* + 1, \perp, 0)$.

Let $p_{j^*,\beta} = \Pr\big[\beta = D(\mathsf{ct}^{(\beta)})\big]$. Recall we have that $\Pr[j \notin S_{\text{indx}} \wedge \mathsf{Diff\text{-}Adv}_j] \geq \delta(\lambda)$. Thus we can write

$$\begin{aligned}
&\Pr[j \notin S \wedge ((p_{j,0} + p_{j,1})/2 - (p_{j+1,0} - p_{j+1,1})/2) \geq \varepsilon/8n_{\text{indx}}] \geq \delta(\lambda) \\
\Rightarrow &\Pr[j^* = j \wedge j \notin S \wedge ((p_{j,0} + p_{j,1})/2 - (p_{j+1,0} - p_{j+1,1})/2) \geq \varepsilon/8n_{\text{indx}}] \geq \delta(\lambda)/n_{\text{indx}}.
\end{aligned}$$

Thus, we can also write that there exists a bit b such that

$$\Pr[j^* = j \wedge j \notin S \wedge (p_{j+1,b} - p_{j,b}) \geq \varepsilon/8n_{\text{indx}}] \geq \delta(\lambda)/n_{\text{indx}}.$$

Now since the reduction algorithm $\mathcal{B}$ simply randomly guesses this bit b, thus we have that with probability at least $\delta(\lambda)/2n_{\text{indx}}$, $\mathcal{B}$ outputs a message $\mathbf{x}^{(b)}$ such that D can distinguish between encryptions of $\mathbf{x}^{(b)}$ to indices j^* and $j^* + 1$ with advantage at least $\varepsilon/8n_{\text{indx}}$. Thus, the lemma follows. □

Next we show that if D output by the adversary $\mathcal{A}$ is a good decoder, then with overwhelming probability the tracing algorithm outputs a non-empty set T. In particular, we show the following lemma.

Lemma 3. *If the scheme* EIPLMCFE *is a 1-query secure EIPLMCFE scheme as per Definition 18 to Definition 22, then for every* PPT *adversary $\mathcal{A}$, polynomial $q(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > 1/q(\lambda)$,*

$$\texttt{Pr-CorrectTr}_{\mathcal{A},\epsilon}(\lambda) \geq \texttt{Pr-GoodDec}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}(\lambda),$$

where $\texttt{Pr-CorrectTr}_{\mathcal{A},\epsilon}(\cdot)$ *and* $\texttt{Pr-GoodDec}_{\mathcal{A},\epsilon}(\cdot)$ *are defined in Definition 16.*

We first state a claim required to prove Lemma 3, and then assume correctness of this claim to prove Lemma 3. In the rest of this section, we prove the required claim, which is the bulk of our proof.

Claim 4.1. *If the scheme* EIPLMCFE *is a 1-query message hiding EIPLMCFE scheme as per Definition 22,*

$$p_{n_{\text{indx}}+1,\perp} \leq \frac{1}{2} + \mathsf{negl}_2(\lambda),$$

for some negligible function $\mathsf{negl}_2(\cdot)$.

Assuming Claim 4.1, we are ready to prove Lemma 3.

Proof of Lemma 3. The proof is almost identical to that of [GKW19, Theorem 5.3]. We first analyze the probability that the tracing algorithm outputs a non-empty set T.

Let $S_{\text{index}} \subseteq [n_{\text{indx}}]$ be the set of indices $j \in [n_{\text{indx}}]$ such that $p_{j,\perp} - p_{j+1,\perp} > \frac{\epsilon}{2n_{\text{indx}}}$. By using a Chernoff bound, we get that

$$\forall\, j \in S_{\text{index}}, \quad \Pr[\hat{p}_{j,\perp} - \hat{p}_{j+1,\perp} < \epsilon/4n_{\text{indx}}] \leq 2^{-\mathcal{O}(\lambda)} = \mathsf{negl}_1(\lambda), \tag{7}$$

where $\hat{p}$ denotes the corresponding estimate computed by the tracing algorithm (i.e., the estimation output by $\mathsf{IndexTrace}$) using $N = \lambda \cdot n_{\text{indx}}/\epsilon$ independent samples.[18]

Furthermore, by definition, when the event $\mathsf{GoodDec}$ happens, we have $p_{\text{nrml}} \geq \frac{1}{2} + \epsilon$ for some non-negligible ϵ. By 1-query normal hiding security, we have that $p_{\text{nrml}} - p_{1,\perp} \leq \mathsf{negl}_3(\lambda)$ for some negligible function $\mathsf{negl}_3(\cdot)$. Combining with Claim 4.1, we have

$$p_{1,\perp} - p_{n_{\text{indx}}+1,\perp} \geq \epsilon - \mathsf{negl}_2(\lambda) - \mathsf{negl}_3(\lambda) \geq \epsilon/2.$$

This allows us to conclude that whenever $\mathsf{GoodDec}$ occurs, S_{index} is a non-empty set. Together with Eq. (7), we have that if $\mathsf{GoodDec}$ occurs, with overwhelming probability the following holds:

$$T^{\text{index}} \neq \emptyset, \quad \mathsf{and} \forall\, (j,p,q) \in T^{\text{index}} : p - q > \frac{\epsilon}{4n_{\text{indx}}}.$$

Notice that the algorithm $\mathsf{IdTrace}$ simply checks for each $\ell \in [\kappa]$, either $p_{j,\ell} > \frac{q+q}{2}$ and sets $\mathsf{id}_\ell = 1$, and $\mathsf{id}_\ell = 0$ otherwise. Thus, for any triple $(j,p,q) \in T^{\text{index}}$, $\mathsf{IdTrace}$ always outputs some identity id. This means that $T^{\text{index}} \neq \emptyset$ implies $T \neq \emptyset$. Therefore, we can write

$$\Pr[T \neq \emptyset] \geq (1 - n_{\text{indx}} \cdot \mathsf{negl}_1(\lambda))\mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) \geq \mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}(\lambda).$$

Finally, combining this with Lemma 1, we obtain

$$\mathtt{Pr\text{-}CorrectTr}_{\mathcal{A},\epsilon}(\lambda) \geq \mathtt{Pr\text{-}GoodDec}_{\mathcal{A},\epsilon}(\lambda) - \mathsf{negl}(\lambda),$$

which concludes the proof. □

Overall, Lemma 1 and Lemma 3 together conclude the proof of Theorem 4.

4.2.4 Proof of Claim 4.1

Proof. Let $\mathcal{A}$ be the adversary in the tracing game defined in Definition 16. We will show that if $p_{n_{\text{indx}}+1,\perp} \geq \frac{1}{2} + \varepsilon$, with respect to some decoder D output by $\mathcal{A}$, for some non-negligible ε, then we can build an adversary $\mathcal{B}$ for the 1-query message-hiding security of $\mathsf{EIPLMCFE}$. The crucial part of the proof is to make our reduction algorithm $\mathcal{B}$ an admissible adversary (see Definition 4) for the message-hiding security game, even though we do not have any restriction on the key queries and the challenge output by the adversary $\mathcal{A}$.

The construction of $\mathcal{B}$ is as follows.

1. **Setup.** Upon receiving the parameters $(1^\kappa, 1^{n_{\text{indx}}})$ from $\mathcal{A}$, $\mathcal{B}$ outputs $(1^\kappa, 1^{n_{\text{indx}}})$ and obtains pp from $(\cdot, \mathsf{pp}, \cdot) \leftarrow \mathsf{EIPLMCFE.Setup}(1^\lambda, 1^n, 1^\kappa, n_{\text{indx}})$. $\mathcal{B}$ generates random PRF keys $\{\mathsf{k}_{i,i'}\}_{i,i' \in [n]}$ as in the first step of Setup and k_{ind} as in the second step of Setup. Then, $\mathcal{B}$ sends pp to $\mathcal{A}$.
   Furthermore, $\mathcal{B}$ also makes n queries to the $\mathsf{EIPLMCFE.Corrupt}$ oracle on different inputs $i \in [n]$ to obtain all encryption keys $(\mathsf{ek}_i)_{i \in [n]}$.
2. **Simulate KeyGen queries.** $\mathcal{B}$ simulates each query to the KeyGen oracle by $\mathcal{A}$ by using k_{ind} and making n queries to its $\mathsf{EIPLMCFE.KeyGen}$ oracle.
3. **Simulate $\mathsf{Corrupt}$ queries.** For each query to the $\mathsf{Corrupt}$ oracle by $\mathcal{A}$ on input $i \in [n]$, $\mathcal{B}$ returns $(\mathsf{ek}_i, \{\mathsf{k}_{i,i'}\}_{i' \in [n]})$ to $\mathcal{A}$. (Note that $\mathcal{B}$ already obtained all encryption keys from the setup phase above, so it needs not to make any queries to its $\mathsf{EIPLMCFE.Corrupt}$ in this step.)

[18] A formal proof follows similarly those provided in [GKW18, Lemma 4.4] and [GKRW18, Lemma 5.3].

4. **Simulate Enc queries.** On the query input $\mathbf{x}_i$ from $\mathcal{A}$, $\mathcal{B}$ answers by computing Enc itself. Note that $\mathcal{B}$ can simulate these encryption queries because it has all PRF keys $\{\mathsf{k}_{i,i'}\}_{i,i' \in [n]}$ as well as encryption keys $(\mathsf{ek}_i)_{i \in [n]}$.
5. **Simulate challenge query.** Whenever $\mathcal{A}$ outputs two challenge message $\mathbf{x}_i^{(\gamma)}$ for $\gamma \in \{0,1\}$, $\mathcal{B}$ computes $\mathbf{w}_i^{(\gamma)}$ as in $\mathsf{TraceEnc}$. Again note that to compute $\mathbf{w}_i^{(\gamma)}$, $\mathcal{B}$ only needs
   - the PRF key $\mathsf{k}_{\mathsf{ind}}$ which it already generated itself;
   - the PRF keys $\{\mathsf{k}_{i,j}\}_{i,j}$ which it obtained from the corrupted encryption keys $(\mathsf{ek}_i)_i$.

   $\mathcal{B}$ outputs $(\mathbf{w}_i^{(0)}, \mathbf{w}_i^{(1)})$ as its challenge plaintexts.
6. **Output.** Upon receiving the challenge ciphertext $(\mathsf{ct}_i^{(b)})_i$ from the $\mathsf{EIPLMCFE}$ challenger, where $\mathsf{ct}_i^{(b)} \leftarrow \mathsf{EIPLMCFE.SplEnc}(\mathsf{msk}, \mathsf{tag}, (n_{\mathrm{indx}}+1, \perp, 0), \mathbf{w}_i^{(b)})$ for $b \leftarrow\!\!\$\ \{0,1\}$, $\mathcal{B}$ runs D on $(\mathsf{ct}_i^b, \hat{\mathbf{v}}_i)_i$ which outputs a bit $b' \in \{0,1\}$, where $\hat{\mathbf{v}}_i$ is some uniformly random vector generated in the previous step during the generation of $\mathbf{w}_i^{(b)}$. Finally, $\mathcal{B}$ outputs the same bit b' as its own guess of b.

We finish the proof by showing that: (i) $\mathcal{B}$ simulates $\mathcal{A}$ perfectly; and (ii) $\mathcal{B}$ is admissible in the message-hiding security game of $\mathsf{EIPLMCFE}$ (Definition 22).

The former is easy to see, except that now $\mathcal{A}$ (or more precisely, the decoder D) now receives a special encryption for its challenge on index $j = n_{\mathrm{indx}} + 1$, instead of a normal encryption. On a special encryption on index $j = n_{\mathrm{indx}} + 1$, $\mathcal{A}$ receives a ciphertext of $\mathsf{EIPLMCFE.SplEnc}$ together with a vector $\hat{\mathbf{v}}_i \coloneqq \mathbf{v}_i^{(\gamma)} + \mathbf{v}_i$ (which plays the same role as the vector $\mathbf{a}_i$ in $\mathsf{EIPLMCFE.Enc}$). However, this vector $\hat{\mathbf{v}}_i$ is one-time padded and perfectly indistinguishable from uniformly random (since $\mathbf{v}_i$ is uniformly random and perfectly hidden from $\mathcal{A}$). Note that without one-time pad encryption, $\mathcal{A}$ can easily detect if it is currently in the tracing mode, by checking the Equations (1)-(2) in $\mathsf{TraceEnc}$ on the vectors $\hat{\mathbf{v}}_i$.[19] This shows that the probability that $\mathcal{B}$ outputs a correct guess is exactly $p_{n_{\mathrm{indx}}+1,\perp}$.

Next, we argue that $\mathcal{B}$ is admissible in the message-hiding security game of $\mathsf{EIPLMCFE}$. For each slot $i \in [n]$, let $\mathbf{z}_i = \mathbf{y} \otimes \mathbf{b}_{j,i}$ be any key query on $\mathbf{y}$ and index j from $\mathcal{A}$. Since $\mathcal{B}$ corrupts all n encryption keys, we need to show that

$$\langle \mathbf{w}^{(0)}_{n_{\mathrm{indx}}+1,i}, \mathbf{z}_i \rangle = \langle \mathbf{w}^{(1)}_{n_{\mathrm{indx}}+1,i}, \mathbf{z}_i \rangle, \ \forall\ i \in [n],$$

which follows from the fact that

$$\begin{aligned} \langle \mathbf{w}^{(\gamma)}_{n_{\mathrm{indx}}+1,i}, \mathbf{z}_i \rangle &= \langle (\mathbf{x}_i^{(\gamma)} + \mathbf{t}_{i,\mathsf{tag}}) \otimes \mathbf{v}_i^{(\gamma)}, \mathbf{y} \otimes \mathbf{b}_{j,i} \rangle \\ &= \langle (\mathbf{x}_i^{(\gamma)} + \mathbf{t}_{i,\mathsf{tag}}), \mathbf{y} \rangle \cdot \langle \mathbf{v}_i^{(\gamma)}, \mathbf{b}_{j,i} \rangle = 0, \end{aligned}$$

where the last equation comes follows from the way $\mathbf{v}_i^{(\gamma)}$ are chosen in Equations (1)-(2).

This concludes our proof. □

5 Embedded Identities Traceable Multi-Client Function: From Indexed to Bounded Schemes

In this section we use our *indexed embedded identities traceable multi-client functional encryption* from previous Section 4 to construct a *bounded embedded identities traceable* MCFE scheme, *as per* Definition 12. The function class is $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ that is defined in Definition 11. Putting forth our ideas, the transformation makes use of an additional *signature scheme*, while developing new ideas different from [GKW19] to handle slot-by-slot secret sharing in the context of multi-client (the foregoing work treats only the case of public key encryption and there is only *one* encryptor).

A crucial step is to obtain a *bounded* EITT-MCFE from the indexed EITT-MCFE (Item **TMCFE.3**). We achieve a generic transformation in the *multi-client* that relies on minimal assumption of the existence

[19] Note that $\mathcal{A}$ can efficiently verify this condition because it has all the vectors $(\mathbf{b}_j)_{j \in [n_{\mathrm{indx}}]}$ from its key queries, in case it corrupts all decryption keys with indices $j \in [n_{\mathrm{indx}}]$.

of signatures, this means our transformation is as efficient as the one in [GKW19] which is in the simpler *single-client* setting[20]. Our transformation is given in Section 5. Under a parameter n_{bd} that bounds the number of functional keys that can be asked under an identity, we use the indexed EITT-MCFE with index space $[2n_{\mathrm{bd}}^2]$ to construct a bounded EITT-MCFE with respect to n_{bd}. The bounded EITT-MCFE generates a key for $\mathbf{y}$ under an identity id by first signing id, concatenating the signature to id to get $\widehat{\mathsf{id}}$ for the indexed EITT-MCFE, and then extracting a functional key for $\mathbf{Y}$ under $\widehat{\mathsf{id}}$ with a random index from $[2n_{\mathrm{bd}}^2]$. To encrypt under a tag tag, the bounded EITT-MCFE secretly shares the message $\mathbf{x}_i$ of a client i into λ shares, λ is the security parameter. Illustratively each i fills in a *column* of a matrix of size $\lambda \times n$ via the encryption under $\mathsf{tag}_j := \mathsf{tag} \oplus j$ of the indexed EITT-MCFE. Then, at decryption time, for each $j \in [\lambda]$ we regroup in a non-interactive way *row by row* the shares of the clients, by decrypting under the tag tag_j each row having n clients. The last computation is a lienar combination of all the row-decryption. As the last but not least challenge, in the IND-CPA proof of our bounded EITT-MCFE, this amplification of the tags tag to tag_j for all $j \in [\lambda]$ means the total keys that must be used is $\lambda n_{\mathrm{bd}} = \Omega(\sqrt{n_{\mathrm{indx}}})$, where $n_{\mathrm{indx}} = 2n_{\mathrm{bd}}^2$ of the indexed EITT-MCFE. Our reduction from bounded to indexed EITT-MCFE, the key queries **KeyGen** from the bounded adversary, in which *no index is repeated*, are handled by a careful sampling of index, then querying for key queries to the indexed EITT-MCFE challenger. Roughly speaking, our sampler has to somehow beat the birthday bound over the index space $[2n_{\mathrm{bd}}^2]$, and we perform the index sampling over small ranges, keeping track of the aready chosen, and increase the ranges over time. As the birthday bound in our case is tightly giving significant collision probablity, we can only achieve an inverse-of-poly success probablity for our sampler (see Lemma 4). This technique fully extends the one for *single-client* in [GKW19] to the *multi-client* setting.

5.1 From Indexed EITT to Bounded EITT

Let $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ be the function class defined in Definition 11. We now present our construction of a *bounded embedded identities traceable multi-client functional encryption* scheme for the function class $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$. Let $\lambda \in \mathbb{N}$ and $\mathcal{ID} = \{0,1\}^\kappa$ be the identity space where $\kappa = \kappa(\lambda)$ is a polynomial in λ. Let $\mathsf{SS} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme over a message space $\mathcal{M} := \mathcal{ID}$ with a signature space $\{0,1\}^{\ell_{\mathsf{sig}}}$, where $\ell_{\mathsf{sig}} = \ell_{\mathsf{sig}}(\lambda)$ is a polynomial in λ, and $\mathsf{EITT\text{-}Idx} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec}, \mathsf{Trace})$ be an indexed EITT-MCFE scheme $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ with identities from $\mathcal{ID}$. We construct a bounded EITT-MCFE scheme $\mathsf{EITT\text{-}Bnd} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Extract}, \mathsf{Dec}, \mathsf{Trace})$ as follows.

$(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}, \mathsf{tk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^\kappa, n_{\mathrm{bd}})$**:** Given as input a security parameter λ, identity space index κ, a number of clients 1^n, and some bound on number of key queries n_{bd}, first run tht set up of $\mathsf{EITT\text{-}Idx}$ to get λ copies of n-client ensemble of keys, that is,

$$\forall\, j \in [\lambda] : (\mathsf{msk}_j^{\mathsf{ei\text{-}idx}}, \mathsf{pp}_j^{\mathsf{ei\text{-}idx}}, (\mathsf{ek}_{j,i}^{\mathsf{ei\text{-}idx}})_{i\in[n]}, \mathsf{tk}_j^{\mathsf{ei\text{-}idx}}) \leftarrow \mathsf{Setup}^{\mathsf{ei\text{-}idx}}(1^\lambda, 1^{\kappa+\ell_{\mathsf{sig}}}, 1^\nu, 1^n)$$

where the index space is of size $\nu := 2 \cdot n_{\mathrm{bd}}^2$. Then, set up the signature keys $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{SS.Setup}(1^\lambda)$. Define and publish the public parameters as $\mathsf{pp} := (\mathsf{pp}_1^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{pp}_\lambda^{\mathsf{ei\text{-}idx}}, \mathsf{vk})$. The master secret key is $\mathsf{msk} := (\mathsf{msk}_1^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{msk}_\lambda^{\mathsf{ei\text{-}idx}}, \mathsf{sk})$, for each client $i \in [n]$ the encryption key is $\mathsf{ek}_i := (\mathsf{ek}_{1,i}^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{ek}_{\lambda,i}^{\mathsf{ei\text{-}idx}})$.

$\mathsf{dk}_{\mathsf{id},\mathbf{Y}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \mathbf{Y})$**:** Given the master secret key msk, an identity $\mathsf{id} \in \{0,1\}^\kappa$, a function description $\mathbf{Y} = \mathbf{y}_1 \| \cdots \| \mathbf{y}_n \in \mathbb{Z}_p^{n\cdot N}$, where each $\mathbf{y}_i \in \mathbb{Z}_p^N$ with respect to $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ and $N = \max_i(N_i)$ with padding of zeroes if need be, first parse msk as $(\mathsf{msk}_1^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{msk}_\lambda^{\mathsf{ei\text{-}idx}}, \mathsf{sk})$. Then, sign the identity id to get a signature $\sigma := \mathsf{Sign}(\mathsf{sk}, \mathsf{id})$. The new identity to pass to the indexed EITT-MCFE scheme is $\widehat{\mathsf{id}} := \mathsf{id} \| \sigma \in \{0,1\}^{\kappa+\ell_{\mathsf{sig}}}$. For each $j \in [\lambda]$, choose an index uniformly at random $\mathsf{idx}_j \xleftarrow{\$} [\nu]$ and run the key generation algorithm

[20] We recall that our EITT-MCFE is under *strong* admissiblity, its IND-CPA covers the case of single-client via recent results [NPP25].

of EITT-Idx to get the decryption key $\mathsf{dk}^{\mathsf{ei\text{-}idx}}_{\widehat{\mathsf{id}},\mathbf{Y},j} \leftarrow \mathsf{KeyGen}^{\mathsf{ei\text{-}idx}}(\mathsf{msk}, \widehat{\mathsf{id}}, \mathsf{idx}_j, \mathbf{Y})$. Return

$$\mathsf{dk}_{\mathsf{id},\mathbf{Y}} := (\mathsf{dk}_{\widehat{\mathsf{id}},\mathbf{Y},1}, \ldots, \mathsf{dk}_{\widehat{\mathsf{id}},\mathbf{Y},\lambda}) \ .$$

$\underline{\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i)}$**:** Given as inputs the public parameters pp, an encryption key ek_i, a message $\mathbf{x}_i \in\in \mathbb{Z}_p^N$, and a tag tag, perform the following steps:

- Parse the public parameters as $(\mathsf{pp}_1^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{pp}_\lambda^{\mathsf{ei\text{-}idx}}, \mathsf{vk})$ and the encryption key as $(\mathsf{ek}_{1,i}^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{ek}_{\lambda,i}^{\mathsf{ei\text{-}idx}})$.
- Sample $\lambda - 1$ random vectors $\mathbf{r}_{j,i} \xleftarrow{\$} \mathbb{Z}_p^N$ for $j \in [\lambda - 1]$ and define $\mathbf{r}_{\lambda,i} := -\sum_{j\in[\lambda-1]} \mathbf{r}_{j,i} + \mathbf{x}_i$. The ensemble $\mathbf{r}_{1,i}, \ldots, \mathbf{r}_{\lambda,i}$ forms a λ-out-of-λ secret sharing of $\mathbf{x}_i$.
- Run the encryption of the underlying EITT-Idx scheme to get

$$\forall\, j \in [\lambda] : \mathsf{ct}_{j,i}^{\mathsf{ei\text{-}idx}} := \mathsf{Enc}^{\mathsf{ei\text{-}idx}}(\mathsf{pp}_j^{\mathsf{ei\text{-}idx}}, \mathsf{ek}_{j,i}^{\mathsf{ei\text{-}idx}}, \widehat{\mathsf{tag}}_j := \mathsf{tag} \oplus j, \mathbf{r}_{j,i})$$

where $\widehat{\mathsf{tag}}_j$ is the tag for the indexed EITT-MCFE scheme.

The ciphertext is $\mathsf{ct}_{\mathsf{tag},i} := (\mathsf{ct}_{1,i}^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{ct}_{\lambda,i}^{\mathsf{ei\text{-}idx}})$.

$\underline{z \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{F_\lambda}, \mathbf{c})}$**:** Given the public parameters pp, a decryption key dk_{F_λ} and a vector of ciphertexts $\mathbf{c} := (\mathsf{ct}_{\mathsf{tag},i})_i$ of length n, first for each $i \in [n]$ parse $\mathsf{ct}_{\mathsf{tag},i}$ as $(\mathsf{ct}_{1,i}^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{ct}_{\lambda,i}^{\mathsf{ei\text{-}idx}})$. The parse the key as $(\mathsf{dk}_{\widehat{\mathsf{id}},\mathbf{Y},1}, \ldots, \mathsf{dk}_{\widehat{\mathsf{id}},\mathbf{Y},\lambda})$. Perform the following steps:

1. For each $j \in [\lambda]$, run the decryption of the underlying EITT-Idx scheme to get

$$\forall\, j \in [\lambda] : \langle \mathbf{r}_{j,1} \| \cdots \| \mathbf{r}_{j,n}, \mathbf{Y} \rangle := \mathsf{Dec}^{\mathsf{ei\text{-}idx}}(\mathsf{pp}_j^{\mathsf{ei\text{-}idx}}, \mathsf{dk}_{\widehat{\mathsf{id}},\mathbf{Y},j}^{\mathsf{ei\text{-}idx}}, (\mathsf{ct}_{j,i}^{\mathsf{ei\text{-}idx}})_{i=1}^n)$$

where $\widehat{\mathsf{id}}$ is the identity with the signature.

2. Compute the output as $z := \sum_{j\in[\lambda]} \langle \mathbf{r}_{j,1} \| \cdots \| \mathbf{r}_{j,n}, \mathbf{Y} \rangle$.

Then output z.

$\underline{T \leftarrow \mathsf{Trace}^D(\mathsf{tk}, 1^y, \mathsf{tag}, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})}$**:** Given oracle access to a program D, the tracing key tk, parameters y, a tag tag and two messages $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$, output a set T of identities where $T \subseteq \{0,1\}^\kappa$.

We call the tracing as public or private depending on whether tk is equal to pp or it is kept secret.

Correctness. Step 1 of the decryption algorithm is correct by the correctness of the underlying EITT-Idx scheme, while decrypting a set of n-client ciphertexts $(\mathsf{ct}_{j,i}^{\mathsf{ei\text{-}idx}})_{i=1}^n$ with respect to a common tag $\widehat{\mathsf{tag}}_j$. The final sum gives

$$\begin{aligned} z = \sum_{j\in[\lambda]} \langle \mathbf{r}_{j,1} \| \cdots \| \mathbf{r}_{j,n}, \mathbf{Y} \rangle &= \langle \sum_{j=1}^{\lambda} \mathbf{r}_{j,1} \| \ldots \| \sum_{j=1}^{\lambda} \mathbf{r}_{j,n}, \mathbf{Y} \rangle \\ &\overset{(*)}{=} \langle \mathbf{x}_1 \| \ldots \| \mathbf{x}_n, \mathbf{Y} \rangle \\ &= \sum_{i=1}^{n} \langle \mathbf{x}_i, \mathbf{y}_i \rangle \ , \end{aligned}$$

where $(*)$ follows from the fact that $\mathbf{r}_{\lambda,i} = -\sum_{j\in[\lambda-1]} \mathbf{r}_{j,i} + \mathbf{x}_i$.

Security. We show the IND-CPA security of the scheme, *as per* Definition 5.

We also note that the tracing security given in [GKW19] is generically applicable to our construction. We thus omit the security proof for tracing.

Theorem 5. *For the class* $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$*, for sufficiently large* $\lambda \in \mathbb{N}$*, suppose EITT-Idx satisfies* adaptive one-challenge *IND-CPA security (Definition 5) with* complete *queries and* repetitions*, under* all-but-two adaptive corruption*, where* all-but-two adaptive corruption *means at most* $(n-2)$*-among-*n *clients can be adaptively corrupted. Then, bounded embedded identities traitor tracing multi-client function scheme EITT-Bnd in Section 5 satisfies* adaptive one-challenge *IND-CPA security with* complete *queries and* repetitions*, under* all-but-two adaptive corruption*.*

Proof of Theorem 5. The strong admissibility condition for the bounded EITT-MCFE scheme is the same as the one for the indexed EITT-MCFE scheme, as it is for the class $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$. We recall that the details are given in Definition 23, and recalled below:

1. For all vectors $(\mathbf{X}^{(0)}, \mathbf{X}^{(1)})$ that is queried to **LoR**, for all $(\mathsf{tag}, \mathbf{Y} = (\mathbf{y}_i)_i^n) \in \mathcal{Q}$, $\sum_{i\in\mathcal{H}} \langle \Delta\mathbf{x}_i, \mathbf{y}_i\rangle = 0$ where $\Delta\mathbf{x}_i = \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(b)}$, where $\mathcal{H} \coloneqq \mathcal{H}_{\mathsf{ekey}}$, for any $b \in \{0,1\}$.
2. For all vectors $(\mathbf{X}^{(0)}, \mathbf{X}^{(1)})$ that is queried to **LoR**, for all $(\mathsf{tag}, \mathbf{Y} = (\mathbf{y}_i)_i^n) \in \mathcal{Q}$, for all $i \in \mathcal{C}_{\mathsf{ekey}}$, we have $\langle \Delta\mathbf{x}_i, \mathbf{y}_i\rangle = 0$.

We perform a direct reduction from the adversary $\mathcal{A}$ that breaks the IND-CPA security of the bounded EITT-MCFE scheme to the adversary $\mathcal{B}$ that breaks the IND-CPA security of the indexed EITT-MCFE scheme. The simulation of the security game for the *bounded* EITT-MCFE, by $\mathcal{B}$ for $\mathcal{A}$, is as follows.

Setup: The adversary $\mathcal{B}$ runs the setup of the signature scheme to get the keys $(\mathsf{sk}, \mathsf{vk})$, as well as the parameters for signature length ℓ_{sig}. Then $\mathcal{B}$ receives the parameters from $\mathcal{A}$ for the *bounded* EITT-MCFE scheme, including $(1^\lambda, 1^\kappa, 1^n, n_{\mathrm{bd}})$. The identities for the *indexed* EITT-MCFE scheme are of length $\kappa + \ell_{\mathsf{sig}}$, $\mathcal{B}$ also defines an index space of size $\nu \coloneqq 2 \cdot n_{\mathrm{bd}}^2$. Then $\mathcal{B}$ asks the IND-CPA challenger of $\mathsf{EITT\text{-}Idx}$ runs the setup of the indexed EITT-MCFE scheme $\mathsf{Setup}^{\mathsf{ei\text{-}idx}}(1^\lambda, 1^{\kappa+\ell_{\mathsf{sig}}}, 1^\nu, 1^n)$ to get the keys $(\mathsf{pp}^{\mathsf{ei\text{-}idx}})$. Then $\mathcal{B}$ defines

$$\mathsf{pp}_1^{\mathsf{ei\text{-}idx}} \coloneqq \mathsf{pp}^{\mathsf{ei\text{-}idx}} \ ,$$

and for $j \in [2;\lambda]$, runs on its own

$$(\mathsf{msk}_j^{\mathsf{ei\text{-}idx}}, \mathsf{pp}_j^{\mathsf{ei\text{-}idx}}, (\mathsf{ek}_{j,i}^{\mathsf{ei\text{-}idx}})_{i\in[n]}, \mathsf{tk}_j^{\mathsf{ei\text{-}idx}}) \leftarrow \mathsf{Setup}^{\mathsf{ei\text{-}idx}}(1^\lambda, 1^{\kappa+\ell_{\mathsf{sig}}}, 1^\nu, 1^n)$$

that is, $\mathcal{B}$ using the first challenge instance of the indexed EITT-MCFE scheme and simulate the remaining $\lambda - 1$. The public parameters are $\mathsf{pp} \coloneqq ((\mathsf{pp}^{\mathsf{ei\text{-}idx}})_{j=1}^\lambda, \mathsf{vk})$, $\mathcal{B}$ then sends to $\mathcal{A}$ the public parameters pp and the master secret key msk. $\mathcal{B}$ also initialize a set $\mathcal{Q}_{\mathsf{KGen}} \coloneqq \varnothing$ to keep track of the indices that will be made to the **Extract** oracle of the *indexed* EITT-MCFE scheme.

Queries: The adversary $\mathcal{B}$ simulates the queries of $\mathcal{A}$ as follows.

- **Extract:** Upon receiving a key query $(\mathsf{id}^{(t)}, \mathbf{Y}^{(t)})$ from $\mathcal{A}$, where the index $t \in [n_{\mathrm{bd}}]$ as $\mathcal{A}$ is restrained to ask at most n_{bd} key queries, $\mathcal{B}$ uses the secret signing key sk to sign the identity $\mathsf{id}^{(t)}$ to get a signature $\sigma^{(t)} \coloneqq \mathsf{Sign}(\mathsf{sk}, \mathsf{id}^t)$. The new identity to pass to the *indexed* EITT-MCFE scheme is $\widehat{\mathsf{id}}^{(t)} \coloneqq \mathsf{id}^{(t)} \| \sigma^{(t)} \in \{0,1\}^{\kappa+\ell_{\mathsf{sig}}}$. For each $j \in [\lambda]$, $\mathcal{B}$ performs the following **Sampling-Idx**$(t, j, \mathcal{Q}_{\mathsf{KGen}})$ procedure:
  - Define $\tilde{\nu} \coloneqq (t-1)\cdot\lambda + j \in \mathbb{N}$.
  - Define functions $\epsilon = \epsilon(\lambda, n_{\mathrm{bd}}), \theta = \theta(\lambda, n_{\mathrm{bd}}), \eta = \eta(\lambda, n_{\mathrm{bd}})$ where $\epsilon, \theta, \eta : \mathbb{N}\times\mathbb{N} \to \mathbb{R}_{>0}$ are such that: for sufficiently large $\lambda \in \mathbb{N}$ and $n_{\mathrm{bd}} \in \mathbb{N}$,

$$\begin{cases} \frac{1}{2}\cdot(\log(\theta\tilde{\nu}) + 1) + \theta\cdot\tilde{\nu} + \log\lambda > 2\cdot\tilde{\nu} \\ \theta = \frac{e}{e-1} - \epsilon > 1 \\ \eta = \left(\frac{e}{e-1} - \epsilon\right)\cdot\frac{\tilde{\nu}}{\nu} \end{cases} \tag{8}$$

  - Sample $\mathsf{idx}_j^{(t)} \xleftarrow{\$} \mathsf{Set}[\eta\nu]$ uniformly at random, repeat if $\mathsf{idx}_j^{(t)} \in \mathcal{Q}_{\mathsf{KGen}}$. We denote here $\mathsf{Set}[\eta\nu]$ a random subset of size $\eta\nu$ of $[\nu]$. In other words, $\mathcal{B}$ samples $\mathsf{idx}_j^{(t)}$ uniformly from $\mathsf{Set}[\eta\nu]$ with rejection, until condition $\mathsf{idx}_j^{(t)} \notin \{\mathsf{idx}_1^{(t)}, \ldots, \mathsf{idx}_{j-1}^{(t)}\} \cup \mathcal{Q}_{\mathsf{KGen}}$.
  - Add $\mathsf{idx}_j^{(t)}$ to $\mathcal{Q}_{\mathsf{KGen}}$.

  $\mathcal{B}$ then queries the *indexed* EITT-MCFE challenger obtain the decryption key $\mathsf{dk\text{-}ididx}^{(t)}_{\widehat{\mathsf{id}},\mathbf{Y},j} \leftarrow \mathsf{KeyGen}^{\mathsf{ei\text{-}idx}}(\mathsf{msk}, \widehat{\mathsf{id}}^{(t)}, \mathsf{idx}_j^{(t)}, \mathbf{Y}^{(t)})$. $\mathcal{B}$ returns to $\mathcal{A}$ the key

$$\mathsf{dk}^{\mathsf{ei\text{-}bnd}}_{\mathsf{id},\mathbf{Y}} \coloneqq (\mathsf{dk\text{-}ididx}^{(t)}_{\widehat{\mathsf{id}},\mathbf{Y},1}, \ldots, \mathsf{dk\text{-}ididx}^{(t)}_{\widehat{\mathsf{id}},\mathbf{Y},\lambda}) \ .$$

Since $\mathcal{B}$ samples idx_j with rejection until getting a new index, we bound the probability that conditioned on fixed (t, j) no collisions happen for the first $t-1$ keys from $\mathcal{A}$ and the first $j-1$ choices of the t-th query, the following event happens:

$\mathsf{idx}_j^{(t)} \notin \{\mathsf{idx}_1^{(t)}, \ldots, \mathsf{idx}_{j-1}^{(t)}\} \cup \mathcal{Q}_{\mathsf{KGen}}$, over the choices of $\mathsf{idx}_j^{(t)} \xleftarrow{\$} [\eta\nu]$.

The following Lemma 4 shows that $\mathcal{B}$ runs in PPT time to simulate the key queries of $\mathcal{A}$. This will make sure $\mathcal{B}$'s *indexed* key queries respect the contrain that no index $\mathsf{idx}_j^{(t)}$ is asked twice over all simulated queries for n_{bd} keys asked by $\mathcal{A}$, and the rejection sampling for $\mathsf{idx}_j^{(t)}$ is PPT.

Lemma 4. *On fixed (t, j) and suppose no collisions happen for the first $t-1$ keys from $\mathcal{A}$ and the first $j-1$ choices of the t-th query, the probability that* **Sampling-Idx**$(t, j, \mathcal{Q}_{\mathsf{KGen}})$ *outputs a fresh $\mathsf{idx}_j^{(t)}$ is at least $1 - 1/\lambda$.*

Proof of Lemma 4. System 8 always has asymptotic solutions for ϵ, θ, η and large enough $\lambda \in \mathbb{N}$ and $n_{\mathrm{bd}} \in \mathbb{N}$ where n_{bd} is a polynomial in λ. This is thanks to the fact that linear function is always asymptotically larger than logarithmic function (so as to solve for $\theta(\lambda, n_{\mathrm{bd}})$). Next, we call "$\mathsf{idx}_j^{(t)} \notin \{\mathsf{idx}_1^{(t)}, \ldots, \mathsf{idx}_{j-1}^{(t)}\} \cup \mathcal{Q}_{\mathsf{KGen}}$" as $\mathsf{NoCollisions}_j^{(t)}$ and "on fixed (t, j) no collisions happen for the first $t-1$ keys from $\mathcal{A}$ and the first $j-1$ choices of the t-th query" as $\mathsf{NoPriorCollisions}_j^{(t)}$. For any fixed (t, j), as $\tilde{\nu} = (t-1) \cdot \lambda + j < \eta\nu$, any subset of size $\eta\nu$ of $[\nu]$ has one fresh $\mathsf{idx}_j^{(t)}$. The following step is to bound the probability that **Sampling-Idx**$(t, j, \mathcal{Q}_{\mathsf{KGen}})$ outputs a fresh $\mathsf{idx}_j^{(t)}$. It is then sufficient to caculate and bound:

$$
\begin{aligned}
&\Pr_{\substack{(t,j)\in[n_{\mathrm{bd}}]\times[\nu] \\ \mathsf{idx}_j^{(t)} \xleftarrow{\$} [\eta\nu]}} \left[\mathsf{NoCollisions}_j^{(t)} | \mathsf{NoPriorCollisions}_j^{(t)}\right] \\
&= \frac{\eta\nu \cdot (\eta\nu - 1) \cdots (\eta\nu - ((t-1)\cdot\lambda + j) + 1)}{\eta\nu^{(t-1)\cdot\lambda + j}} \\
&= \frac{(\eta\nu)!}{((\eta\nu) - (t-1)\lambda - j)! \cdot (\eta\nu)^{(t-1)\cdot\lambda + j}} \\
&\overset{(*)}{\geq} \frac{\sqrt{2\pi\eta\nu}\left(\frac{\eta\nu}{e}\right)^{\eta\nu} \cdot \exp(\frac{1}{12\eta\nu+1})}{\sqrt{2\pi(\eta\nu - \tilde{\nu})}\left(\frac{\eta\nu-\tilde{\nu}}{e}\right)^{\eta\nu-\tilde{\nu}} \cdot \exp(\frac{1}{12(\eta\nu-\tilde{\nu})})} \cdot \frac{1}{(\eta\nu)^{\tilde{\nu}}} \\
&\qquad \text{where } \tilde{\nu} \coloneqq (t-1)\cdot\lambda + j \\
&\overset{(**)}{>} \left(\frac{\eta\nu}{\eta\nu - \tilde{\nu}}\right)^{1/2} \cdot \left(\frac{(\eta\nu)^{\eta\nu}}{\exp(\tilde{\nu}) \cdot (\eta\nu - \tilde{\nu})^{\eta\nu-\tilde{\nu}}}\right) \cdot \frac{1}{(\eta\nu)^{\tilde{\nu}}} \\
&= \left(\frac{\eta\nu}{e \cdot (\eta\nu - \tilde{\nu})}\right)^{\eta\nu-\tilde{\nu}+1/2} \cdot \frac{1}{\exp(2\tilde{\nu} - \eta\nu - 1/2)} \ ,
\end{aligned}
$$

where $(*)$ follows from Stirling's approximation

$$
\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot \exp\left(\frac{1}{12n+1}\right) < n! < \sqrt{2\pi n}\left(\frac{n}{e}\right)^n \cdot \exp\left(\frac{1}{12n}\right)
$$

and $(**)$ follows from the fact that $\frac{\exp(\frac{1}{12\eta\nu+1})}{\exp(\frac{1}{12(\eta\nu-\tilde{\nu})})} > 1$. In order for

$$
\Pr_{\substack{(t,j)\in[n_{\mathrm{bd}}]\times[\nu] \\ \mathsf{idx}_j^{(t)} \xleftarrow{\$} [\eta\nu]}} \left[\mathsf{NoCollisions}_j^{(t)} | \mathsf{NoPriorCollisions}_j^{(t)}\right] \geq \frac{1}{\lambda}
$$

it suffices to verify that the existing asymptotic solutions for ϵ, θ, η from Equation (8) satisfy the system of inequalities given n_{bd}, λ:

$$\begin{cases} \eta\nu \geq \frac{e}{e-1} \cdot \tilde{\nu} \text{ where } \tilde{\nu} = (t-1) \cdot \lambda + j \leq n_{\mathrm{bd}}\lambda \\ \exp(2\tilde{\nu} - \eta\nu - 1/2) \leq (\eta\nu)^{1/2} \cdot \lambda \ . \end{cases}$$

The first inequality is satisfied by the choice of η and ϵ such that $\eta = \left(\frac{e}{e-1} - \epsilon\right) \cdot \frac{\tilde{\nu}}{\nu}$ in Equation (8). The second inequality is satisfied by the choice of η, ϵ and θ such that:

$$\frac{1}{2} \cdot (\log(\theta\tilde{\nu}) + 1) + \theta \cdot \tilde{\nu} + \log\lambda > 2 \cdot \tilde{\nu}$$
$$\theta = \frac{e}{e-1} - \epsilon > 1 \ .$$

We emphasize that we only need asymptotic solutions for ϵ, θ, η and large enough $\lambda \in \mathbb{N}$ and $n_{\mathrm{bd}} \in \mathbb{N}$. The proof is concluded. □

- **Encrypt:** Upon receiving an encryption query $(\mathsf{tag}, i, \mathbf{x}_i)$ from $\mathcal{A}$,
  - Sample $\lambda - 1$ random vectors $\mathbf{r}_{j,i} \xleftarrow{\$} \mathbb{Z}_p^N$ for $j \in [\lambda - 1]$ and define $\mathbf{r}_{\lambda,i} := -\sum_{j\in[\lambda-1]} \mathbf{r}_{j,i} + \mathbf{x}_i$. The ensemble $\mathbf{r}_{1,i}, \ldots, \mathbf{r}_{\lambda,i}$ forms a λ-out-of-λ secret sharing of $\mathbf{x}_i$.
  - For $j = 1$, $\mathcal{B}$ queries **Enc** of the *indexed* EITT-MCFE challenger to get the ciphertexts $\mathsf{ct}_{1,i}^{\mathsf{ei\text{-}idx}}$ for $(\mathsf{ek}_{1,i}^{\mathsf{ei\text{-}idx}}, \widehat{\mathsf{tag}}_j := \mathsf{tag} \oplus 1, \mathbf{r}_{1,i})$.
  - For $j = 2, \ldots, \lambda$ Run the encryption of the underlying $\mathsf{EITT\text{-}Idx}$ scheme to get

$$\forall\, j \in [\lambda] : \mathsf{ct}_{j,i}^{\mathsf{ei\text{-}idx}} := \mathsf{Enc}^{\mathsf{ei\text{-}idx}}(\mathsf{pp}_j^{\mathsf{ei\text{-}idx}}, \mathsf{ek}_{j,i}^{\mathsf{ei\text{-}idx}}, \widehat{\mathsf{tag}}_j := \mathsf{tag} \oplus j, \mathbf{r}_{j,i})$$

  where $\widehat{\mathsf{tag}}_j$ is the tag for the indexed EITT-MCFE scheme.

  The ciphertext is $\mathsf{ct}_{\mathsf{tag},i} := (\mathsf{ct}_{1,i}^{\mathsf{ei\text{-}idx}}, \ldots, \mathsf{ct}_{\lambda,i}^{\mathsf{ei\text{-}idx}})$.
- **Corruption:** Upon receiving a corruption query i from $\mathcal{A}$, $\mathcal{B}$ corrupts $\mathsf{ek\text{-}ididx}_{1,i}$ from its *indexed* EITT-MCFE challenger. Then $\mathcal{B}$ takes the $\lambda - 1$ set up keys $\mathsf{ek\text{-}ididx}_{j,i}$ for $j \in [2; \lambda]$ and returns $(\mathsf{ek\text{-}ididx}_{1,i}, \ldots, \mathsf{ek\text{-}ididx}_{\lambda,i})$ to $\mathcal{A}$.

Challenge: Upon receiving a challenge $(\mathsf{tag}^*, \mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ from $\mathcal{A}$, $\mathcal{B}$ performs as in the **Encrypt** query to get the ciphertexts

$$\mathsf{ct}_{\mathsf{tag},i}^{(b)} := (\mathsf{ct}_{1,i}^{\mathsf{ei\text{-}idx},b}, \ldots, \mathsf{ct}_{\lambda,i}^{\mathsf{ei\text{-}idx},b}) \ .$$

The first component $\mathsf{ct}_{1,i}^{\mathsf{ei\text{-}idx},b}$ is obtained from the *indexed* EITT-MCFE challenger, while the rest are obtained by the set up parameters by $\mathcal{B}$. It remains to verify the admissibility:

- Over honest i, it holds that, for each $j \in [\lambda - 1]$:

$$\langle \sum_{i\in\mathcal{H}} \mathbf{r}_{j,i}^{(b)}, \mathbf{y}_i \rangle = \langle \sum_{i\in\mathcal{H}} \mathbf{r}_{j,i}^{(b)}, \mathbf{y}_i \rangle \ ,$$

  because by construction the $\lambda - 1$ random vectors $\mathbf{r}_{j,i}^{(b)} \xleftarrow{\$} \mathbb{Z}_p^N$ for $j \in [\lambda - 1]$ are independent of the challenge $\mathbf{x}_i^{(b)}$.
- Over honest i, it holds that, for $j = \lambda$:

$$\langle \sum_{i\in\mathcal{H}} \mathbf{r}_{\lambda,i}^{(b)}, \mathbf{y}_i \rangle = \langle \sum_{i\in\mathcal{H}} \mathbf{r}_{\lambda,i}^{(b)}, \mathbf{y}_i \rangle \ ,$$

  thanks to the fact that $\mathbf{r}_{\lambda,i}^{(b)} = -\sum_{j\in[\lambda-1]} \mathbf{r}_{j,i}^{(b)} + \mathbf{x}_i^{(b)} = \mathsf{const} + \mathbf{x}_i^{(b)}$.
- Over corrupted i, it is worth noting that each corrupted ek_i reveals the whole set of ciphertexts $\mathsf{ct}_{\mathsf{tag},i}$, *i.e.* $\mathsf{ek\text{-}ididx}_{j,i}$ for all $j \in [\lambda]$ are revealed. Thus we do *not* need to verify for individual $j \in [\lambda]$, meanwhile for individual $i \in [n]$ the admissibility for corrupted i follows the admissibility of $\mathcal{A}$.

Finally, when $\mathcal{A}$ outputs a guess b', $\mathcal{B}$ outputs b' as well.

It can be verified from the above simulation that $\mathcal{B}$ runs in PPT and if $\mathcal{A}$ wins the IND-CPA game for the bounded EITT-MCFE scheme, then $\mathcal{B}$ wins the IND-CPA game for the indexed EITT-MCFE scheme. The proof is concluded. □

6 From Weak Admissibility to Strong Admissibility for MCFE for Inner Products with Access Control

We present a transformation to turn an MCFE that is provably secure under the *weak* admissiblity condition (Definition 3) into an MCFE that is provably secure under the *strong* admissibility condition (Definition 4), for the function class $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} \times \mathsf{AC\text{-}K}$. Let $\mathcal{E}^{\mathsf{w}} = (\mathsf{Setup}^{\mathsf{w}}, \mathsf{Extract}^{\mathsf{w}}, \mathsf{Enc}^{\mathsf{w}}, \mathsf{Dec}^{\mathsf{w}})$ be an MCFE that is provably secure under the *weak* admissiblity condition for the function class $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} \times \mathsf{AC\text{-}K}$ ($\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}$ is defined in Definition 11). We remark that the function class captures computation of sums of inner products of vectors, where the vectors are indexed by clients $i \in [n]$, together with an access control via AC-K. We require the following properties for $\mathcal{E}^{\mathsf{w}}$:

1. The functional key of $\mathcal{E}^{\mathsf{w}}$ can be decomposed $\mathsf{dk}^{\mathsf{w}}_{\mathbf{Y},\mathsf{ac\text{-}k}} = (\widetilde{\mathsf{dk}}^{\mathsf{w}}_i)_i$ for $i \in [n]$.
2. $\mathcal{E}^{\mathsf{w}}$ satisfies a form of decryption that:

$$\mathcal{E}^{\mathsf{w}}.\mathsf{Dec}(\mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}}, (\mathsf{ct}_i)_{i=1}^n) = \mathsf{decode}\left(\sum_{i=1}^{n} \mathsf{PartialDec}(\widetilde{\mathsf{dk}}^{\mathsf{w}}_i, \widetilde{\mathsf{ct}}^{\mathsf{w}}_i)\right)$$

   where $\mathsf{PartialDec}(\widetilde{\mathsf{dk}}^{\mathsf{w}}_i, \widetilde{\mathsf{ct}}^{\mathsf{w}}_i) = \mathsf{encode}\left(\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i \rangle + \mathit{share}_i\right)$, encode is homomorphic under $+$ and $\mathsf{decode}(\mathsf{encode}(\cdot))$ is identity only over a polynomially large interval, and share_i is an n-out-of-n share of 0, or maybe some value T fixed and used only for decryption (*e.g.* the case of [LT19]).

The usage of encode and decode resembles the *two-step decryption* requirement in the transformation from FE to MIFE in [ACF+18], later treated in the multi-user setting in [ABKW19]. We denote by $\mathcal{E}^{\mathsf{als\text{-}ip}} = (\mathsf{Setup}^{\mathsf{als\text{-}ip}}, \mathsf{Extract}^{\mathsf{als\text{-}ip}}, \mathsf{Enc}^{\mathsf{als\text{-}ip}}, \mathsf{Dec}^{\mathsf{als\text{-}ip}})$ a generic (single-client) public key IPFE scheme, *i.e.,* for $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_1}$. We transform $\mathcal{E}^{\mathsf{w}}$ into another MCFE $\mathcal{E} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ for the same function class $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} \times \mathsf{AC\text{-}K}$ as in Figure 7.

6.1 Security

In the following Theorem 6 we prove the *adaptively one-challenge (with private-input repetitions) IND-security* of the transformed MCFE $\mathcal{E}$, under *dynamic* corruption and *selective* functions $(\mathbf{y}_i)_i$ with *adaptive/selective* access control ac-k, assuming the *adaptively one-challenge (with private-input repetitions) IND-security* of $\mathcal{E}^{\mathsf{w}}$, under the same constraints, and the *adaptive SIM-security* of $\mathcal{E}^{\mathsf{als\text{-}ip}}$.

Theorem 6. *The MCFE scheme* $\mathcal{E} = (\mathsf{Setup}, \mathsf{DKeyGen}, \mathsf{Enc}, \mathsf{Dec})$ *is* one-challenge (with private-input repetitions) IND-CPA secure*, under* dynamic *corruption and* semi-adaptive *queries (adaptive challenges with* selective *key queries for functions* $(\mathbf{y}_i)_i$*) with* adaptive/selective *access control* ac-k *(Definition 5), if* $\mathcal{E}^{\mathsf{w}}$ *is one-challenge (with private-input repetitions) IND-secure under the same constraint and* $\mathcal{E}^{\mathsf{als\text{-}ip}}$ *is SIM-secure.*

Remark 6. We note that for SIM-security of $\mathcal{E}^{\mathsf{als\text{-}ip}}$, we can instantiate using a SIM-secure (variant of) LWE-based IPFE by Agrawal, Libert, and Stehlé [ALS16], that is proved one-challenge SIM-secure in [ALMT20]. Moreover, the transformation produces a MCFE that is secure under the *strong* admissibility condition, resilient to *dynamic* corruption, *adaptive* challenges with *selective* functions $(\mathbf{y}_i)_i$, and *adaptive* access control ac-k. The selective function constraint is due to the usage of the simulator of $\mathcal{E}^{\mathsf{als\text{-}ip}}$. Putting forth our final traitor tracing MCFE scheme, this semi-adaptive IND-CPA under strong admissibility suffices so that any

$\underline{\mathsf{Setup}(1^\lambda)}$**:** Run

$$\begin{cases} \mathcal{E}^{\mathsf{w}}.\mathsf{Setup}(1^\lambda) = \mathsf{msk}^{\mathsf{w}}, (\mathsf{ek}_i^{\mathsf{w}})_{i=1}^n \\ \mathcal{E}^{\mathsf{als\text{-}ip}}.\mathsf{Setup}^{\mathsf{als\text{-}ip}}(1^\lambda, 1^{N_i}) = (\mathsf{msk}_i, \mathsf{pk}_i) \text{ for each } i \end{cases}.$$

Sample $\mathbf{v}_i \xleftarrow{\$} \mathbb{Z}_q^{N_i}, \mathbf{t}_i \xleftarrow{\$} (\mathbb{Z}_q^*)^{N_i}$ and output the encryption keys as well as the secret keys as follows:

$$\begin{cases} \mathsf{ek}_i & = (\mathsf{pk}_i^{\mathsf{als\text{-}ip}}, \mathsf{ek}_i^{\mathsf{w}}, \mathbf{t}_i, \mathbf{v}_i) \\ \mathsf{msk} & = ((\mathsf{msk}_i^{\mathsf{als\text{-}ip}})_{i=1}^n, \mathsf{msk}^{\mathsf{w}}, (\mathbf{v}_i[k]/\mathbf{t}_i[k])_{k=1}^{N_i}) \end{cases}.$$

$\underline{\mathsf{Extract}(\mathsf{msk}, (\mathbf{y}_i)_{i=1}^n, \mathsf{ac\text{-}k})}$**:** The parameters are $\mathbf{y}_i \in \mathbb{Z}_q^{N_i}$ defining a function to compute inner products with respect to $\mathbf{Y} = [\mathbf{y}_1 \parallel \ldots \parallel \mathbf{y}_n]$. Parse the master secret key

$$\mathsf{msk} = ((\mathsf{msk}_i^{\mathsf{als\text{-}ip}})_{i=1}^n, \mathsf{msk}^{\mathsf{w}}, (\mathbf{v}_i[k]/\mathbf{t}_i[k])_{k=1}^{N_i})$$

and compute

$$\mathsf{dk}^{\mathsf{w}}_{\mathbf{Y},\mathsf{ac\text{-}k}} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{Extract}(\mathsf{msk}^{\mathsf{w}}, (\mathbf{y}_i)_{i=1}^n, \mathsf{ac\text{-}k})$$

then for each i, generate $\mathsf{dk}_i^{\mathsf{als\text{-}ip}} \leftarrow \mathcal{E}^{\mathsf{als\text{-}ip}}.\mathsf{Extract}(\mathsf{msk}_i^{\mathsf{als\text{-}ip}}, (\frac{\mathbf{v}_i[k]\cdot\mathbf{y}_i[k]}{\mathbf{t}_i[k]})_{k=1}^{N_i})$. Output $\mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}} = (\mathsf{dk}^{\mathsf{w}}_{\mathbf{Y},\mathsf{ac\text{-}k}}, (\mathsf{dk}_i^{\mathsf{als\text{-}ip}})_i)$.

$\underline{\mathsf{Enc}(\mathsf{ek}_i, \mathsf{tag}, \mathbf{x}_i \in \mathbb{Z}_q^{N_i}, \mathsf{ac\text{-}ct}_i)}$**:** Given a vector $\mathbf{x}_i \in \mathbb{Z}_q^{N_i}$, and public attributes $\mathsf{ac\text{-}ct}_i$, parse the encryption key $\mathsf{ek}_i = (\mathsf{pk}_i^{\mathsf{als\text{-}ip}}, \mathsf{ek}_i^{\mathsf{w}}, \mathbf{t}_i, \mathbf{v}_i)$. Then, sample $r_i \xleftarrow{\$} \mathbb{Z}_q$ and compute

$$\mathsf{ct}_i^{\mathsf{w}} \leftarrow \mathcal{E}^{\mathsf{w}}.\mathsf{Enc}(\mathsf{ek}_i^{\mathsf{w}}, \mathsf{tag}, \mathbf{x}_i - r_i \cdot \mathbf{v}_i, \mathsf{ac\text{-}ct}_i)$$

and $\mathsf{ct}_i^{\mathsf{als\text{-}ip}} \leftarrow \mathcal{E}^{\mathsf{als\text{-}ip}}.\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{als\text{-}ip}}, r_i \cdot \mathbf{t}_i)$. Output the ciphertext $\mathsf{ct}_i = (\mathsf{ct}_i^{\mathsf{w}}, \mathsf{ct}_i^{\mathsf{als\text{-}ip}})$.

$\underline{\mathsf{Dec}(\mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}}, (\mathsf{ct}_i)_{i=1}^n)}$**:** Parse $\mathsf{dk}_{\mathbf{Y},\mathsf{ac\text{-}k}} = (\mathsf{dk}^{\mathsf{w}}_{\mathbf{Y},\mathsf{ac\text{-}k}}, (\mathsf{dk}_i^{\mathsf{als\text{-}ip}})_i)$ and $\mathsf{ct}_i = (\mathsf{ct}_i^{\mathsf{w}}, \mathsf{ct}_i^{\mathsf{als\text{-}ip}})$. Compute

$$\begin{aligned} \mathsf{out}^{\mathsf{w}} &= \sum_{i=1}^n \mathsf{PartialDec}(\mathsf{dk}_i^{\mathsf{w}}, \mathsf{ct}_i^{\mathsf{w}}) = \sum_{i=1}^n \mathsf{encode}\left(\langle \mathbf{x}_i - r_i\mathbf{v}_i, \mathbf{y}_i\rangle + share_i\right) \\ &= \mathsf{encode}(\sum_{i=1}^n \langle \mathbf{x}_i - r_i\mathbf{v}_i, \mathbf{y}_i\rangle) \end{aligned}$$

We also compute

$$\begin{aligned} \mathsf{out}^{\mathsf{als\text{-}ip}} = \sum_{i=1}^n \mathsf{encode}(\mathcal{E}^{\mathsf{als\text{-}ip}}.\mathsf{Dec}(\mathsf{dk}_i^{\mathsf{als\text{-}ip}}, \mathsf{ct}_i^{\mathsf{als\text{-}ip}})) &= \sum_{i=1}^n \mathsf{encode}(\langle r_i\mathbf{v}_i, \mathbf{y}_i\rangle) \\ &= \mathsf{encode}(\sum_{i=1}^n \langle r_i\mathbf{v}_i, \mathbf{y}_i\rangle) \end{aligned}.$$

In the end we compute

$$\mathsf{decode}\left(\mathsf{out}^{\mathsf{w}} + \mathsf{out}^{\mathsf{als\text{-}ip}}\right) = \mathsf{decode}\left(\mathsf{encode}(\sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i\rangle)\right) = \sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i\rangle$$

due to properties of the function class that $\sum_{i=1}^n \langle \mathbf{x}_i, \mathbf{y}_i\rangle$ lies in a polynomially large interval.

Fig. 7. The transformation of an MCFE with weak admissibility into an MCFE with strong admissibility.

tracing advaersry $\mathcal{A}$ can still query *adaptive* functions $(\mathbf{y}_i)_i$ up to the point of construting its pirate decoder (Definition 22): it is only when the decoder is defined (hence all the queried functions are known) we will reduce to the security of the MCFE for $\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n} \times \mathsf{AC\text{-}K}$ with strong admissibility.

Proof Of Theorem 6. The advantage of an adversary $\mathcal{A}$ in a game G_i is denoted by

$$\mathsf{Adv}(\mathsf{G}_i) \coloneqq |\Pr[\mathsf{G}_i = 1] - 1/2|$$

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

Game G_0: This is the *one-challenge (with repetitions)* security game, with *strong* admissibility. The private-input repetitions at each position $i \in [n]$ are indexed by $\mathsf{rep} \in [J_i]$ where J_i is the maximum repetitions queried for position i. We note that for different i, the bound J_i can be different. The challenge ciphertext encrypts subvectors $\mathbf{x}_i^{(b,\mathsf{rep})} \in \mathbb{Z}_q^N$. For simplicity, we add a constraint that the challenge tag tag is *not* queried to **Enc**. This incurs a multiplicative loss factor in advantage up to an inverse of polynomial in λ, where we can reduce to the normal 1chal by guessing the challenge tag among the tags for encryption, and responding all of its **Enc** queries $(i, \mathbf{x}_i, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$ by $\mathbf{LoR}(i, \mathbf{x}_i, \mathbf{x}_i, (\mathsf{tag}, \mathsf{ac\text{-}ct}_i))$.

Game G_1: If $\Delta\mathbf{x}_i = \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(1)} = 0$, our simulator encrypts $\mathbf{x}_i^{(0)}$ using $\mathbf{LoR}^{\mathsf{w}}$ of $\mathcal{E}^{\mathsf{w}}$ to simulate the partial challenge ciphertexts. There are two cases:

- If later i is corrupted, the admissibility condition 2 of $\mathcal{E}^{\mathsf{w}}$ is satisfied by $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{(1)}$, and this gives us a reduction to the IND-security of $\mathcal{E}^{\mathsf{w}}$.
- Else, i stays honest and the admissibility condition 1 of $\mathcal{E}^{\mathsf{w}}$ is satisfied by $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{(1)}$ and the fact that the admissibility condition of $\mathcal{E}$ is also satisfied over the sum of $i \in \mathcal{H}$ (condition 1), which also gives us a reduction to the IND-security of $\mathcal{E}^{\mathsf{w}}$.

The change is bounded by $\mathsf{Adv}^{\mathsf{mc\text{-}pos\text{-}ind\text{-}1chal\text{-}cpa}}_{\mathcal{E}^{\mathsf{w}},\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}\times\mathsf{AC\text{-}K}}(1^\lambda)$.

Game G_2: We treat the case $\Delta\mathbf{x} \neq \mathbf{0}$ by programming the master secret $(\mathbf{t}_i, \mathbf{v}_i)$ and switching from $b \xleftarrow{\$} \{0,1\}$ to 0. When receiving $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ and $\mathbf{x}_i^{(0)} \neq \mathbf{x}_i^{(1)}$, the simulator implicitly replaces $\mathbf{v}_i$ in the ciphertext challenge query to its oracle $\mathbf{LoR}^{\mathsf{w}}$ of $\mathcal{E}^{\mathsf{w}}$ by

$$\widetilde{\mathbf{v}}_i \coloneqq \mathbf{v}_i - \frac{1}{r_i}\Delta\mathbf{x}_i$$

where $\Delta\mathbf{x}_i \coloneqq \mathbf{x}_i^{(0)} - \mathbf{x}_i^{(1)}$ is known at the time of simulating ciphertext challenge queries. Moreover, r_i is chosen at the time of encryption to query $\mathbf{LoR}$ of $\mathcal{E}^{\mathsf{w}}$. Since $\mathbf{v}_i \xleftarrow{\$} \mathbb{Z}_q^{N_i}$ and $r_i \xleftarrow{\$} \mathbb{Z}_q$, the newly programmed $\widetilde{\mathbf{v}}_i$ has the same distribution as that of $\mathbf{v}_i$. Concretely, the pair of challenge queries to the $\mathbf{LoR}^{\mathsf{w}}$ of $\mathcal{E}^{\mathsf{w}}$ is now

$$(\mathbf{x}_i^{(0)} - r_i\mathbf{v}_i, \mathbf{x}_i^{(1)} - r_i\widetilde{\mathbf{v}}_i) = (\mathbf{x}_i^{(0)} - r_i\mathbf{v}_i, \mathbf{x}_i^{(0)} - r_i\mathbf{v}_i) \tag{9}$$

and the returned $\mathcal{E}^{\mathsf{w}}$ ciphetext does *not* depend on b anymore. Finally, using the simulator of $\mathcal{E}^{\mathsf{als\text{-}ip}}$ to simulate the parts $\mathsf{ct}_i^{(\mathsf{als\text{-}ip},chal)}$ of the challenge ciphertexts, for encrypting $\mathcal{E}^{\mathsf{sim\text{-}als}}.\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{sim\text{-}als}}, r_i \cdot \mathbf{t}_i)$ using their simuatled key $\mathsf{pk}_i^{\mathsf{sim\text{-}als}}$. The challenges to return the adversary are $(\mathsf{ct}_i^{(\mathsf{w},chal)}, \mathsf{ct}_i^{(\mathsf{als\text{-}ip},chal)})$. We consider case by case:

- Necessarily the changes on $\mathbf{v}_i$ into $\widetilde{\mathbf{v}}_i$ stay indistinguishable under the one-challenge SIM-security of $\mathcal{E}^{\mathsf{als\text{-}ip}}$, using the simulated $\mathsf{Sim}^{\mathsf{als\text{-}ip}}.\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{sim\text{-}als}}, r_i \cdot \mathbf{t}_i)$, knowing the evaluations $\mathsf{encode}(\langle r_i\mathbf{v}_i, \mathbf{y}_i\rangle)$ due to the *selective* function constraint on the adversary.
- For the keys that $\mathsf{ac\text{-}k}$ is satisfied $\mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{ac\text{-}ct}_i)_i) = 1$:
  - If i is corrupted, the weak admissibility for corrupted i is ensured by (9).
  - In any cases, the strong admissibility condition 4 of $\mathcal{E}$ ensures $\langle\Delta\mathbf{x}_i, \mathbf{y}_i\rangle = 0$ and correctness is ensured by $\mathcal{E}^{\mathsf{w}}.\mathsf{Dec} = \mathsf{decode}\left(\sum_{i=1}^n \mathsf{PartialDec}(\widetilde{\mathsf{dk}}_i^{\mathsf{w}}, \widetilde{\mathsf{ct}}_i^{\mathsf{w}})\right)$. We recall that $\mathsf{PartialDec}(\mathsf{dk}_i^{\mathsf{w}}, \mathsf{ct}_i^{\mathsf{w}}) = \mathsf{encode}\left(\langle\mathbf{x}_i^{(0)}, \mathbf{y}_i\rangle + share_i\right)$ and the share $share_i$ of 0 will absorb $\langle\Delta\mathbf{x}_i, \mathbf{y}_i\rangle$.

- For the keys that $\mathsf{ac\text{-}k}$ is *not* satisfied $\mathsf{Rel}(\mathsf{ac\text{-}k}, (\mathsf{ac\text{-}ct}_i)_i) = 0$. It is potentially that $\langle \Delta\mathbf{x}_i, \mathbf{y}_i \rangle \neq \mathbf{0}$. Thanks to the *private-input* repetitions, there exists one *unique set* of attributes $\mathsf{ac\text{-}ct}_i$ that corresponds to $\Delta\mathbf{x}_i \neq \mathbf{0}$, and moreover corresponds to the unique $\mathsf{ct}_i^{(\mathsf{w},chal)}$. This allows us to rely on the *private-input* repetitions security of $\mathcal{E}^{\mathsf{w}}$.

The difference in advantages is $\mathsf{Adv}^{\mathsf{mc\text{-}pos\text{-}ind\text{-}1chal\text{-}cpa}}_{\mathcal{E}^{\mathsf{w}},\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_n}\times\mathsf{AC\text{-}K}}(1^\lambda) + \mathsf{Adv}^{\mathsf{sim\text{-}1chal}}_{\mathcal{E}^{\mathsf{als\text{-}ip}},\mathcal{F}^{\mathsf{ip}}_{N_1,\ldots,N_1}}(1^\lambda)$.

At the end of G_2, all ciphertext are encrypted with respect to $\mathbf{x}_i^{(0)}$, independently of b. □

7 Building EI-PLMCFE from Mixed FE and Attribute-Based MCFE

We generalize the approach in [GKW18,CVW+18,GKW19] and construct an EI-PLMCFE scheme from a mixed functional encryption scheme and a key-policy ABMCFE scheme as follows. Let $\mathsf{ABMCFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ be a *multi-client functional encryption (MCFE) scheme with fine-grained access control* (Definition 2) for the function class $\mathcal{F} \times \mathsf{AC\text{-}K}$ (Definition 1). Let $\mathsf{mixedFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{SKEnc}, \mathsf{Dec})$ be a mixed functional encryption scheme for a function class $\mathcal{F}^{\mathsf{mfe}} = \{f^{\mathsf{mfe}}_{\kappa(\lambda)}\}_{\lambda\in\mathbb{N}}$, message space $\{\mathcal{I}_\kappa\}_{\lambda\in\mathbb{N}}$, and ciphertexts of length $\ell(\kappa,\lambda)$, where $\kappa = \kappa(\lambda)$ is a polynomial in λ meanwhile $\ell(\kappa,\lambda)$ is a polynomial in κ and λ. Let $\mathcal{ID} = \{0,1\}^{\log(\kappa)}$ be the identity space whose size is $\kappa = \kappa(\lambda)$. We make the following conventions:

- Fixing $\lambda \in \mathbb{N}$, for each index size $\nu = \nu(\lambda)$ that is polynomially large in λ, we choose the polynomial $\kappa = \kappa(\nu)$ and the set $\mathcal{I}_\kappa$ such that there exists an injective mapping $\{0,1\}^{\log(\nu)+\log(\kappa)} \hookrightarrow \mathcal{I}_\kappa$.
- The function class $\mathcal{F}^{\mathsf{mfe}}$ contains "comparison" ($>$) operators between bitstrings of length $\log(\nu)$ and bit-checking for identities:

$$\mathcal{F}^{\mathsf{mfe}} = \left\{ f^{\mathsf{mfe}}_{\kappa(\nu)} : \{0,1\}^{\log(\nu)} \times \{0,1\}^{\log(\nu)} \times ([\log(\kappa)] \cup \{\perp\}) \times \{0,1\} \times \mathcal{ID} \to \{0,1\} \right\}$$
$$\text{and } f^{\mathsf{mfe}}_{\kappa(\nu)}(j,j',\ell,b,\mathsf{id}) = 1 \iff j' > j \text{ OR } (j,\ell) = (j',\perp) \text{ OR } (j',\mathsf{id}_\ell) = (j,1-b) \ . \tag{10}$$

  When j,ℓ,b are fixed, we write "$f^{\mathsf{mfe}}_{\kappa(\nu),j,\ell,b}(j',\mathsf{id})$" to denote the function evaluation $f^{\mathsf{mfe}}_{\kappa(\nu)}(j,j',\ell,b,\mathsf{id})$.
- Fixing $\lambda \in \mathbb{N}$, for each index size $\nu = \nu(\lambda)$ polynomially large in λ, for $\kappa = \kappa(\nu)$ chosen as above, for each $\ell(\kappa,\lambda)$ polynomial in κ and λ, we choose the polynomial $\tilde{\kappa} = \tilde{\kappa}(\lambda,\kappa)$ such that for each $i \in [\nu]$ there exists an injective mapping $\{0,1\}^{\ell(\kappa,\lambda)} \hookrightarrow \mathsf{AC\text{-}Ct}_{\tilde{\kappa},i}$. We recall that the set of attributes $\mathsf{AC\text{-}Ct}_{\tilde{\kappa},i}$ comes from the definition of $\mathcal{F} \times \mathsf{AC\text{-}K}$ (Definition 1) (where the parameter index $\tilde{\kappa}$ is implicit).
- The relation associated to $\mathcal{F} \times \mathsf{AC\text{-}K}$ is defined as follows: $\mathsf{Rel} : \mathsf{AC\text{-}K} \times \mathsf{AC\text{-}Ct}_1 \times \cdots \times \mathsf{AC\text{-}Ct}_\nu \to \{0,1\}$ where

$$\begin{cases} \mathsf{AC\text{-}K} \supseteq \{\mathsf{mixedFE.Dec}(\mathsf{dk}^{\mathsf{mfe}}_j, \cdot) \ : \ \lambda, n, \nu, j \in [\nu], \mathsf{id} \in \mathcal{ID}, \\ \qquad\qquad \text{and } \mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j} \leftarrow \mathsf{mixedFE.Extract}(\mathsf{msk}^{\mathsf{mfe}}, (j,\mathsf{id}))\} \\ \mathsf{Rel}(\mathsf{ac\text{-}k}, \mathsf{ac\text{-}ct}_1, \ldots, \mathsf{ac\text{-}ct}_\nu) = \prod_{i=1}^{\nu} \mathsf{mixedFE.Dec}(\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j}, \mathsf{ac\text{-}ct}_i) \end{cases} . \tag{11}$$

  We recall that the set $\mathsf{AC\text{-}K}$ and the relation Rel come from the definition of $\mathcal{F} \times \mathsf{AC\text{-}K}$ (Definition 1).

On Obtaining EI-PLMCFE. First of all, the transformation to obtain an embedded identities private linear broadcast tracing for MCFE (Item **TMCFE.2**) in its own is non-trivial due to the requirement of powerful primitive such as the MCFE for attribute-based inner products with enhanced security, combining with a general-purpose mixed FE. For this former instantiation we already have to dedicate an independent section Section 1.2.2 to develop our new security bootstrapping technique. Using our transformation in Section 7, from a LWE-based MCFE for inner products with access control, combining with a strong enough LWE-based mixed FE for bounded depth circuits from [CVW+18], we can obtain an EI-PLMCFE from LWE for computing inner products.

7.1 Construction

Our construction for EI-PLMCFE is given below.

$(\mathsf{msk}, \mathsf{pp}, (\mathsf{ek}_i)_{i\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, 1^\kappa, 1^\nu, 1^n)$**:** Given as input the security parameter λ, the identity space parameter κ, and an index size ν, output a master secret key msk, public parameters pp, and $n = n(\lambda)$ encryption keys $(\mathsf{ek}_i)_{i\in[n]}$ by running the following:

- $(\mathsf{msk}^{\mathsf{mfe}}, \mathsf{pp}^{\mathsf{mfe}}) \leftarrow \mathsf{mixedFE.Setup}(1^\lambda, 1^\kappa)$.
- $(\mathsf{msk}^{\mathsf{abmc}}, \mathsf{pp}^{\mathsf{abmc}}, (\mathsf{ek}_i^{\mathsf{abmc}})_{i\in[n]}) \leftarrow \mathsf{ABMCFE.Setup}(1^\lambda, 1^n, 1^{\tilde{\kappa}})$, where $1^{\tilde{\kappa}}$ additionally specifies the length of ciphertexts chosen as above.

Output

$$\begin{cases} \mathsf{msk} & = (\mathsf{msk}^{\mathsf{mfe}}, \mathsf{msk}^{\mathsf{abmc}}, (\mathsf{ek}_i^{\mathsf{abmc}})_{i=1}^n) \\ \mathsf{pp} & = (\mathsf{pp}^{\mathsf{mfe}}, \mathsf{pp}^{\mathsf{abmc}}) \\ \mathsf{ek}_i & = \mathsf{ek}_i^{\mathsf{abmc}} \text{ for } i \in [n] \end{cases} .$$

$\mathsf{dk}_{(\mathsf{id},j),F_\lambda} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, j, F_\lambda)$**:** Given as input the master secret key msk, an identity $\mathsf{id} \in \{0,1\}^\kappa$, an index $j \in [\nu]$, and a function description $F_\lambda \in \mathcal{F}$, generate the decryption for the *indexed embedded-identity private linear broadcast* aspect combiningly with the functional aspect as follows:

- Given $j \in [\nu]$ and $\mathsf{id} \in \mathcal{ID}$, generate a $\mathsf{mixedFE}$ key $\mathsf{dk}_{\mathsf{id},j}^{\mathsf{mfe}} \leftarrow \mathsf{mixedFE.Extract}(\mathsf{msk}^{\mathsf{mfe}}, (j, \mathsf{id}))$.
- Then, the ABMCFE is used to generate the decryption key on the policy:

$$\mathsf{dk}_{(\mathsf{id},j),F_\lambda}^{\mathsf{abmc}} \leftarrow \mathsf{ABMCFE.Extract}\left(\mathsf{msk}^{\mathsf{abmc}}, \left(F_\lambda, \mathsf{mixedFE.Dec}(\mathsf{dk}_{\mathsf{id},j}^{\mathsf{mfe}}, \cdot)\right)\right) ,$$

  where $\mathsf{mixedFE.Dec}(\mathsf{dk}_{\mathsf{id},j}^{\mathsf{mfe}}, \cdot)$ plays the role of the policy and is a function that takes a $\mathsf{mixedFE}$ ciphertext and decrypts *as per* $\mathsf{mixedFE}$ to output a bit.

Then, output $\mathsf{dk}_{(\mathsf{id},j),F_\lambda} \coloneqq \mathsf{dk}_{(\mathsf{id},j),F_\lambda}^{\mathsf{abmc}}$.

$\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, x_i)$**:** Given as inputs the public parameters pp, an encryption key ek_i, a tag tag, a message $x_i \in \mathcal{D}_{\lambda,i}$, first run the *public* $\mathsf{mixedFE}$ encryption and output a ciphertext $\mathsf{mixedFE.Enc}(\mathsf{pp}^{\mathsf{mfe}}) \to \mathsf{ct}_{\mathrm{pub}}^{\mathsf{mfe}}$. Then, run and output the usual ciphertext of the ABMCFE scheme $\mathsf{ABMCFE.Enc}(\mathsf{pp}^{\mathsf{abmc}}, \mathsf{ek}_i, \widetilde{\mathsf{tag}} \coloneqq (\mathsf{tag}, (0, \perp, 0)), (x_i, \mathsf{ct}_{\mathrm{pub}}^{\mathsf{mfe}})) \to \mathsf{ct}_{\mathsf{tag},i}$, where $\widetilde{\mathsf{tag}}$ is the tag for the ABMCFE scheme (specifying index 0 and not considering the identity's positions with respect to $(\perp, 0)$).

$\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, (\mathsf{tag}, (j, \ell, b)), x_i)$**:** Given as inputs the master secret key msk, a tag tag, a triple $(j, \ell, b) \in [\nu + 1] \times ([\log(\kappa)] \cup \{\perp\}) \times \{0,1\}$, a message $x_i \in \mathcal{D}_{\lambda,i}$, where ι is the size of the identity space $\mathcal{ID}$, first run the $\mathsf{mixedFE}$ encryption on the function $f_{\kappa(\nu),j,\ell,b}^{\mathsf{mfe}} : \mathcal{ID} \times \{0,1\}^{\log(\nu)} \to \{0,1\}$ to obtain a ciphertext $\mathsf{mixedFE.SKEnc}(\mathsf{msk}^{\mathsf{mfe}}, f_{\kappa(\nu),j,\ell,b}^{\mathsf{mfe}}) \to \mathsf{ct}_{j,\ell,b}^{\mathsf{mfe}}$. Then, run the ABMCFE encryption on the message x_i with the tag $\widehat{\mathsf{tag}} \coloneqq (\mathsf{tag}, (j, \ell, b))$, having the attribute $\mathsf{ct}_{j,\ell,b}^{\mathsf{mfe}} \in \mathsf{AC\text{-}Ct}_{\tilde{\kappa},i}$ by set up, and output $\mathsf{ABMCFE.Enc}(\mathsf{pp}^{\mathsf{abmc}}, \mathsf{ek}_i, \widehat{\mathsf{tag}}, (x_i, \mathsf{ct}_{j,\ell,b}^{\mathsf{mfe}})) \to \mathsf{ct}_{\mathsf{tag},i}$.

$z \leftarrow \mathsf{Dec}(\mathsf{pp}, \mathsf{dk}_{\mathsf{id},j,F_\lambda}, \mathbf{c})$**:** Given the public parameters pp, a decryption key $\mathsf{dk}_{\mathsf{id},j,F_\lambda}$ and a vector of ciphertexts $\mathbf{c} \coloneqq (\mathsf{ct}_{\mathsf{tag},i})_i$ of length n, run and output $z \leftarrow \mathsf{ABMCFE.Dec}(\mathsf{dk}_{\mathsf{id},j,F_\lambda}, \mathbf{c})$ where $z \in \mathcal{R}_\lambda \cup \{\perp\}$.

Correctness. The correctness of the scheme follows from the correctness of the underlying mixed functional encryption and ABMCFE schemes. First of all decrypting by ABMCFE controls the decryption following (11):

- (Normal encryption) For a fixed same tag, for each $i \in [n]$, for a normal ciphertext $\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{Enc}(\mathsf{pp}, \mathsf{ek}_i, \mathsf{tag}, x_i)$, the attribute is a public-key $\mathsf{mixedFE}$ ciphertext on the trivial all-1 function, which upon correct $\mathsf{mixedFE}$ decryption will output 1 on all $(\mathsf{id}, j) \in \mathcal{ID} \times [\nu]$ (by (11)), In other words, for any i, the attribute of a normal ciphertext $\mathsf{ct}_{\mathsf{tag},i}$ is always accept by the policy $\mathsf{mixedFE.Dec}(\mathsf{dk}_{\mathsf{id},j}^{\mathsf{mfe}}, \cdot)$ embedded in the ABMCFE key. Since the same $\widetilde{\mathsf{tag}} \coloneqq (\mathsf{tag}, (0, \perp, 0))$ is used, the correctness of ABMCFE decryption allows recover the function evaluation $F_\lambda(x_1, \ldots, x_n)$.

- (Special encryption) Similarly, for a fixed same tag, for each $i \in [n]$, for $(j, \ell, b) \in \{0,1\}^{\log(\nu)} \times ([\log(\kappa)] \cup \{\perp\}) \times \{0,1\}$, for a special ciphertext $\mathsf{ct}_{\mathsf{tag},i} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, (\mathsf{tag}, (j, \ell, b)), x_i)$, the attribute is a $\mathsf{mixedFE}$ ciphertext on the function $f^{\mathsf{mfe}}_{\kappa(\nu),j,\ell,b}$, which upon correct $\mathsf{mixedFE}$ decryption will output 1 *as per* Equation (11) on (id, j') where $j' > j$ or $(j, \ell) = (j', \perp)$ or $(j', \mathsf{id}_\ell) = (j, 1-b)$. The $\mathsf{mixedFE}$ key $\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j}$ is generated for each (id, j'), and the policy $\mathsf{mixedFE.Dec}(\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j}, \cdot)$ that represents a correct $\mathsf{mixedFE}$ decryption, is embedded in the ABMCFE key. Finally, the correctness of ABMCFE decryption allows recover the function evaluation $F_\lambda(x_1, \ldots, x_n)$.

7.2 Security Proof

We now show the 1-bounded security of the EI-PLMCFE scheme.

Theorem 7 (1-query Security). *Let* $\mathit{ABMCFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ *be a* multi-client functional encryption (MCFE) scheme with fine-grained access control *(Definition 2) for the function class* $\mathcal{F} \times \mathsf{AC\text{-}K}$ *(Definition 1). Let* $\mathcal{ID} = \{0,1\}^{\log(\kappa)}$ *be the identity space whose size is* $\kappa = \kappa(\lambda)$. *Let* $\mathit{mixedFE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{SKEnc}, \mathsf{Dec})$ *be a* mixed functional encryption *scheme for a function class* $\mathcal{F}^{\mathit{mfe}} = \{f^{\mathit{mfe}}_{\kappa(\lambda)}\}_{\lambda \in \mathbb{N}}$ *and* $f^{\mathit{mfe}}_{\kappa(\lambda)}$ *is defined* as per (10) *for identities in* $\mathcal{ID}$ *and indices in* $[\nu(\lambda)]$, *message space* $\{\mathcal{I}_\kappa\}_{\lambda \in \mathbb{N}}$, *and ciphertexts of length* $\ell(\kappa, \lambda)$,

Suppose ABMCFE *is* one-challenge selective-attribute IND-CPA secure, *with* private-input repetitions, *and* $\mathit{mixedFE}$ *is* 1-SKEnc function indistinguishability secure *and* 1-SKEnc accept indistinguishability secure. *Then, the EI-PLMCFE scheme from Section 7.1 is* 1-query Normal Hiding *(Definition 18),* 1-query Index Hiding *(Definition 19),* 1-query Upper/Lower Identity Hiding *(Definition 20/Definition 21), and* 1-query Complete Message Hiding *(Definition 22) for the class* $\mathcal{F}$ *with embedded identities from* $\mathcal{ID}$ *and indices from* $[\nu]$.

Proof for Theorem 7. We present the main ideas of the proof, adding details at places that we deem more technical.

1-query Normal Hiding. We prove that the EIPLMCFE from Section 7.1 is 1-*query Normal Hiding* (Definition 18). We perform a reduction to the 1-SKEnc accept indistinguishability security of the underlying $\mathsf{mixedFE}$ scheme.

- **Setup.** Our simulator runs

$$(\mathsf{msk}^{\mathsf{abmc}}, \mathsf{pp}^{\mathsf{abmc}}, (\mathsf{ek}^{\mathsf{abmc}}_i)_{i\in[n]}) \leftarrow \mathsf{ABMCFE.Setup}(1^\lambda, 1^n, 1^{\tilde{\kappa}}) \ .$$

  It then interacts with the challenger for the 1-SKEnc accept indistinguishability game, receiving the public parameters $\mathsf{pp}^{\mathsf{mfe}}$. The nomal hiding adversary then gets $\mathsf{pp} := (\mathsf{pp}^{\mathsf{abmc}}, \mathsf{pp}^{\mathsf{mfe}})$ and $(\mathsf{ek}_i)_{i\in[n]} := (\mathsf{ek}^{\mathsf{abmc}}_i)_{i\in[n]}$.
- **Query Phase.** The normal hiding adversary makes queries to the key generation of the EI-PLMCFE scheme: given a such query $(\mathsf{id}, j, F_\lambda)$, our simulator asks (id, j) to the challenger for the $\mathsf{mixedFE}$ key $\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j}$ then uses the set up $\mathsf{msk}^{\mathsf{abmc}}$ to simulate:

$$\mathsf{dk}^{\mathsf{abmc}}_{(\mathsf{id},j),F_\lambda} \leftarrow \mathsf{ABMCFE.Extract}\left(\mathsf{msk}^{\mathsf{abmc}}, \left(F_\lambda, \mathsf{mixedFE.Dec}(\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j}, \cdot)\right)\right) \ .$$

  Moreover, the single SplEnc query is simulated by: given a such query $(\mathsf{tag}, (j, \ell, b)), x_i$, our simulator asks the challenger for the $\mathsf{mixedFE}$ encryption $\mathsf{mixedFE.SKEnc}(\mathsf{msk}^{\mathsf{mfe}}, f^{\mathsf{mfe}}_{\kappa(\nu),j,\ell,b}) \to \mathsf{ct}^{\mathsf{mfe}}_{j,\ell,b}$, then uses the set up $\mathsf{ek}^{\mathsf{abmc}}_i$ to simulate:

$$\mathsf{ABMCFE.Enc}(\mathsf{pp}^{\mathsf{abmc}}, \mathsf{ek}_i, \widehat{\mathsf{tag}}, (x_i, \mathsf{ct}^{\mathsf{mfe}}_{j,\ell,b})) \to \mathsf{ct}_{\mathsf{tag},i} \ .$$

  The simulation is done uniformly before and after seeing the challenge from the normal hiding adversary.

- **Challenge.** Upon receiving (tag, x_i) from the normal hiding adversary, our simulator forwards a challenge query $f^{(*)}$ to the challenger for the 1-SKEnc accept indistinguishability game, by defining

$$f^{(*)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),1,\perp,0} : \mathcal{ID} \times \{0,1\}^{\log(\nu)} \to \{0,1\}$$

  and $f^{\mathsf{mfe}}_{\kappa(\nu),1,\perp,0}$ behaves *as per* Equation (10), and $f^{(*)} = f^{\mathsf{mfe}}_{\kappa(\nu),1,\perp,0}$ is indeed an all-1 function for all allowed key queries that have $\mathsf{id} \in \mathcal{ID}$ and $j \in [\nu]$. When obtaining the challenge ciphertext ct^*, our simulator forwards it to the normal hiding adversary.
- **Guess.** The normal hiding adversary outputs a guess bit b'. Our simulator outputs b' as well.

It can be inspected that the advantage our simulator has in the 1-SKEnc accept indistinguishability game is the same as the advantage of the normal hiding adversary in the EI-PLMCFE scheme.

1-query Index Hiding. We now prove the 1-*query Index Hiding* (Definition 19) property of the EI-PLMCFE scheme. We perform a reduction to the 1-SKEnc function indistinguishability security of the underlying $\mathsf{mixedFE}$ scheme. The simulation of **Setup** and **Query Phase** is the same as in the 1-query Normal Hiding proof. For **Challenge**, we remark that the challenges forwarded to the function indistinguishability challenger consist of: the challenge index $j^* \in \{0,1\}^{\log(\nu)}$ is output by the index hiding adversary,

$$f^{(0)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),j^*,\perp,0} : \mathcal{ID} \times \{0,1\}^{\log(\nu)} \to \{0,1\} \tag{12}$$

$$f^{(1)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),j^*+1,\perp,0} : \mathcal{ID} \times \{0,1\}^{\log(\nu)} \to \{0,1\} \tag{13}$$

and these satisfy $f^{(0)}(\mathsf{id}, j) = f^{(1)}(\mathsf{id}, j) = 1$ for all allowed key queries to EI-PLMCFE (that incurs a key query to the function indistinguishability challenger of $\mathsf{mixedFE}$) that have $\mathsf{id} \in \mathcal{ID}$ and $j \neq j^*$. We insist that the particular case of $j = j^* + 1$ is covered in the definition of the $\mathsf{mixedFE}$ function class (*c.f.* Equation (10)): $f^{(0)}(\cdot, j^*+1) = 1$ because $j^* + 1 > j^*$ and $f^{(1)}(\cdot, j^*+1) = 1$ because $f^{(1)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),j^*+1,\perp,0}$. The conclusion of the proof is the same as in the 1-query Normal Hiding proof.

1-query Upper/Lower Identity Hiding. We prove the 1-*query Upper/Lower Identity Hiding* (Definition 20/Definition 21) properties of the EI-PLMCFE scheme. We demonstrate the proof for the *Upper Identity Hiding* property, the *Lower Identity Hiding* property is proven similarly. The proof for the *Upper Identity Hiding* property is a reduction to the 1-SKEnc function indistinguishability security of the underlying $\mathsf{mixedFE}$ scheme. The simulation of **Setup** and **Query Phase** is the same as in the 1-query Index Hiding proof. For **Challenge**, we remark that the challenges forwarded to the function indistinguishability challenger consist of: the challenge (index, identity's position, bit-value) $(j^*, \ell^*, b^*) \in \{0,1\}^{\log(\nu)} \in \{0,1\}^{\log(\kappa)} \times \{0,1\}$ is output by the index hiding adversary,

$$f^{(0)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),j^*+1,\perp,0} : \mathcal{ID} \times \{0,1\}^{\log(\nu)} \to \{0,1\}$$

$$f^{(1)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),j^*,\ell^*,b^*} : \mathcal{ID} \times \{0,1\}^{\log(\nu)} \to \{0,1\}$$

and these satisfy $f^{(0)}(\mathsf{id}, j) = f^{(1)}(\mathsf{id}, j)$ for all allowed key queries to EI-PLMCFE (that incurs a key query to the function indistinguishability challenger of $\mathsf{mixedFE}$) that do *not* satisfy (id, j^*) and $\mathsf{id}_{\ell^*} = 1 - b^*$. It suffices to verify that the two functions equals on (id, j^*), using Equation (10): $f^{(0)}(\mathsf{id}, j^*) = 0$ and $f^{(1)}(\mathsf{id}, j^*) = 0$ because the identity's position $\ell^* \neq \perp$ on the same input index j^* but the constraint on key queries $\mathsf{id}_{\ell^*} = b^*$ not meeting the bitchecking of $f^{(1)} \coloneqq f^{\mathsf{mfe}}_{\kappa(\nu),j^*,\ell^*,b^*}$. The conclusion of the proof is the same as in the 1-query Index Hiding proof.

1-query Complete Message Hiding. We prove the 1-*query Complete Message Hiding* (Definition 22) property of the EI-PLMCFE scheme. It is for this property that we need the *one-challenge selective-attribute IND-CPA security* of the underlying ABMCFE scheme, with *private-input repetitions*, *i.e.* fixing a tag and $i \in [n]$, only the message x_i can be queried multiple times to the ABMCFE challenger. The main steps are given below. We perform a reduction to we need the *one-challenge selective-attribute IND-CPA security* of

the underlying ABMCFE scheme, with *private-input repetitions*. The challenge ciphertexts by SplEnc are now computed as follows: by setting $\widehat{\mathsf{tag}} := (\mathsf{tag}, (\nu+2, \perp, 0))$

$$\mathbf{c}^{(b)} := \left(\mathsf{ct}_i^{(b)}\right)_{i=1}^{n}; \mathsf{ct}_i^{(b)} \leftarrow \mathsf{SplEnc}(\mathsf{msk}, \mathsf{tag}, \nu+1, \perp, 0, x_i^{(b)}) \text{ proceeds by } \begin{cases} \mathsf{mixedFE.SKEnc}(\mathsf{msk}^{\mathsf{mfe}}, f^{\mathsf{mfe}}_{\kappa(\nu),\nu+1,\perp,0}) \to \mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\perp,0} \\ \mathsf{ABMCFE.Enc}(\mathsf{pp}^{\mathsf{abmc}}, \mathsf{ek}_i^{\mathsf{abmc}}, \widehat{\mathsf{tag}}, (x_i^{(b)}, \mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\perp,0})) \to \mathsf{ct}_i^{(b)} \end{cases}.$$

Our simulator is against the security of ABMCFE. The simulation is as follows:

- **Setup.** The simulator runs the mixedFE set up to obtain $(\mathsf{msk}^{\mathsf{mfe}}, \mathsf{pp}^{\mathsf{mfe}}) \leftarrow \mathsf{mixedFE.Setup}(1^\lambda, 1^\kappa)$. Then it computes the challenge attributes using mixedFE parameters:

  $$\mathsf{mixedFE.SKEnc}(\mathsf{msk}^{\mathsf{mfe}}, f^{\mathsf{mfe}}_{\kappa(\nu),\nu+1,\perp,0}) \to \mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\perp,0}$$

  and send $\mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\perp,0}$ to the ABMCFE challenger as the *selective-attribute* for challenge. We remark that $f^{\mathsf{mfe}}_{\kappa(\nu),\nu+1,\perp,0}(\mathsf{id}, j) = 0$ for all allowed key queries (id, j) to EI-PLMCFE (that incurs a key query to the ABMCFE challenger), where $j \in [\nu]$ as restrained in Definition 22. Our simulator then receives the public parameters $\mathsf{pp}^{\mathsf{abmc}}$, returns $\mathsf{pp} := (\mathsf{pp}^{\mathsf{abmc}}, \mathsf{pp}^{\mathsf{mfe}})$ to the 1-query message hiding adversary.
- **Query phase.** The queries to **Corrupt** are forwarded to the ABMCFE challenger as the $\mathsf{ek}_i := \mathsf{ek}_i^{\mathsf{abmc}}$ by construction. Others are treated below:
  - For KeyGen oracle: on query id, j, our simulator uses the mixedFE sercret key to

    $$\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j} \leftarrow \mathsf{mixedFE.Extract}(\mathsf{msk}^{\mathsf{mfe}}, (j, \mathsf{id})) \ .$$

    Then it queries to the ABMCFE challenger for key extraction of

    $$\mathsf{dk}^{\mathsf{abmc}}_{(\mathsf{id},j),F_\lambda} \leftarrow \mathsf{ABMCFE.Extract}\left(\mathsf{msk}^{\mathsf{abmc}}, \left(F_\lambda, \mathsf{mixedFE.Dec}(\mathsf{dk}^{\mathsf{mfe}}_{\mathsf{id},j}, \cdot)\right)\right)$$

    and returns $\mathsf{dk}_{(\mathsf{id},j),F_\lambda} := \mathsf{dk}^{\mathsf{abmc}}_{(\mathsf{id},j),F_\lambda}$ to the 1-query message hiding adversary.
  - For the only SplEnc query $(\mathsf{tag}, (\nu+1, \ell, b), x_i)$ by the 1-query message hiding adversary, our simulator uses the set mixedFE parameters to compute $\mathsf{mixedFE.SKEnc}(\mathsf{msk}^{\mathsf{mfe}}, f^{\mathsf{mfe}}_{\kappa(\nu),\nu+1,\ell,b}) \to \mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\ell,b}$. Then it queries to the ABMCFE challenger the ABMCFE encryption of $\widehat{\mathsf{tag}}, (x_i, \mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\ell,b})$.
  - For the simulation of the Enc oracle: our simulator can use the public-key encryption of the mixedFE scheme to first simulate $\mathsf{mixedFE.Enc}(\mathsf{pp}^{\mathsf{mfe}}) \to \mathsf{ct}^{\mathsf{mfe}}_{\mathrm{pub}}$ and queries to the ABMCFE challenger the ABMCFE encryption using $\mathsf{ek}_i^{\mathsf{abmc}}$, obtains the ABMCFE ciphertext and respond as Enc oracle to the message hiding adversary.
- **Challenge.** When the message hiding adversary outputs $\left(D, \mathsf{tag}, \left(x_i^{(0)}, x_i^{(1)}\right)_{i\in[n]}\right)$, our simulator forwards the challenge $\left(x_i^{(0)}, x_i^{(0)}\right)_{i\in[n]}$ to the ABMCFE challenger, knowing the *selective* attribute $\mathsf{ct}^{\mathsf{mfe}}_{\nu+1,\perp,0}$ is declared up front. This change goes indistinguishable thanks to:
  - The private-input repetitions of the ABMCFE scheme, combining with the fact that the mixedFE encryption of $f^{\mathsf{mfe}}_{\kappa(\nu),\nu+1,\perp,0}$ is used as the challenge attribute. This latter is an all-0 function for all allowed key queries to KeyGen (that incurs a key query to the ABMCFE challenger).
  - Next, since no keys allow decryption, and no new attributes are allowed to be queried on the challenge tag (to switch policy control to 1), the change is indistinguishable thanks to the *one-challenge selective-attribute IND-CPA security* of the ABMCFE scheme.

  We note that we do use the *admissibility* property to ensure for any $i \in \mathcal{C}$, the corrupted ek_i do not trivially distinguish given as challenge $\left(D, \mathsf{tag}, \left(x_i^{(0)}, x_i^{(1)}\right)_{\underline{i\in\mathcal{H}}}\right)$. In particular, even thought the $\mathcal{C}$ slots are controlled by the adversary, given function evaluations for keys from KeyGen that might decrypts these corrupted slots on some F_λ, the strong admissibility (Definition 4) makes sure those evaluations do

not trivially affect the $i \in \mathcal{H}$. Finally, the admissibility over $\mathcal{H}$ ensures security vacuously since no keys decrypt the challenge.

The conclusion follows. □

8 Attribute-based Multi-client FE for Inner-Product

The last missing piece in our construction is an AB-MCFE scheme for inner-product. Our AB-MCFE for inner-product needs to satisfy the following requirements:

R.1 Support a specific access control where all clients have the same attribute, evaluated over the same policy function. This is enough for our application of building EI-PLMCFE in Section 7, as the attributes are chosen by the reduction.

R.2 Adaptive corruption, which follows from the fact that the adversary in our tracing model can corrupt any client adaptively before outputting a pirate decoder.

R.3 Security against semi-adaptive chosen functions, selective chosen attribute and adaptive chosen policies, which again due to the construction given in Section 7.

R.4 Security against an adversary who can ask polynomially-many number of challenge queries on arbitrary challenge plaintext pairs and arbitrary tag. In the IND-CPA security game of MCFE, this is captured as allowing repetition on both private-inputs (i.e., the plaintexts) and public-inputs (i.e., the tags) in the admissibility condition. We refer the reader to Section 2.1 for the definition of repetition in the IND-CPA security game of MCFE. This is due to the fact that in our tracing definitions (Definition 12 and Definition 15), the adversary has no restriction on what it can ask to the KeyGen and Enc oracle before outputting its pirate decoder, along with a pair of challenge plaintexts and a tag. We note that this is stronger that what is typically considered in the IND-CPA security for MCFE, where the adversary is not allowed to learn encryption of plaintexts under its challenge tag.[21]

Requirements (**R.1**) and (**R.2**) can be obtained using known techniques. Even though we do not know how to achieve requirement (**R.4**) in the most general case, our main observation is to use requirement (**R.1**) to reduce requirement (**R.4**) to a weaker one with private-inputs repetition. Our idea is very simple: repetition queries are only useful when the decryption key is decryptable (i.e., the attribute of the ciphertext satisfies the policy of the decryption key). Thus, we can consider the attribute as a part of the tag for each encryption. The adversary is then forced to use the same attribute and the same tag for each *complete* challenge query. This effectively prevents the adversary from taking the advantages of mix-and-match attacks.

Consequently, we only need to consider the weak security with private-inputs only repetition, where the adversary can ask many challenge queries for a single tag. In this section, we show how to construct an AB-MCFE for inner-product satisfying these requirements. In particular, we give a construction of key-policies AB-MCFE for circuits $f : \{0,1\}^\ell \to \{0,1\}$ of depth d and size s. We call a key dk_f a 1-key with respect to an attribute $\mathbf{u}$ if $f(\mathbf{u}) = 1$, and a 0-key otherwise. In our language, a 0-key allows to open the ciphertext, but a 1-key does not. We note that in this section, we change our notation and use n as a part of the public parameter of the scheme, instead of the number of clients, and use k to denote the number of clients.

8.1 Lattice Preliminaries

Throughout this section, let $n, m, q \in \mathbb{N}$ such that $n = \mathsf{poly}(\lambda)$ and $m \geq n\lceil \log q \rceil$. We define $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ to be the gadget matrix as the matrix of rank n obtained by padding $\mathbf{I}_n \otimes (1, 2, 4, \ldots, 2^{\lceil \log q \rceil})$ with zero-columns. For any $t \in \mathbb{Z}$, we define $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times t} \to \{0,1\}^{m \times t}$ to be the deterministic algorithm that inputs a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times t}$ and outputs a binary matrix $\mathbf{G}^{-1}(\mathbf{A}) \in \{0,1\}^{m \times t}$ such that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$. Let $D_{\mathbb{Z},\tau}$ denote the discrete Gaussian distribution over $\mathbb{Z}$ with standard deviation τ.

[21] We also note that achieving the strong security notion with full repetition (both on plaintexts and tags) is still open, known constructions achieve private-inputs repetition only.

Lattice trapdoors and preimage sampling. In the following, consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For all $\mathbf{v} \in \mathbb{Z}_q^n$, we let $\mathbf{A}_\tau^{-1}(\mathbf{v})$ denote the random variable whose distribution is $D_{\mathbb{Z},\tau}^m$ conditioned on $\mathbf{A} \cdot \mathbf{A}_\tau^{-1}(\mathbf{v}) = \mathbf{v}$. A τ-trapdoor for $\mathbf{A}$ is a procedure that can sample from a distribution within 2^n statistical distance of $\mathbf{A}_\tau^{-1}(\mathbf{v})$ in time $\mathsf{poly}(n, m, \log q)$, for any $\mathbf{v} \in \mathbb{Z}_q^n$. We slightly overload notation and denote a τ-trapdoor for $\mathbf{A}$ by $\mathbf{A}_\tau^{-1}$. The following properties had been established in a long sequence of works [GPV08, CHKP10, ABB10a, ABB10b, MP12, BLP+13].

Lemma 5 (Properties of Lattice Trapdoors).

- *Given $\mathbf{A}_\tau^{-1}$, it is efficient to generate a trapdoor $[\mathbf{A}\|\mathbf{B}]_{\tau'}^{-1}$ for all $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, for all $m \in \mathbb{N}$ and $\tau' \geq \tau$.*
- *There exists an efficient procedure $\mathsf{TrapGen}(1^n, 1^m, q)$ that outputs $(\mathbf{A}, \mathbf{A}_{\tau_0}^{-1})$ where $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for some $m = \mathcal{O}(n \log q)$ and is 2^n-close to uniform, where $\tau_0 = \omega(\sqrt{n \log q \log m})$.*

Lemma 6 (Preimage Sampling [MP12, GPV08, LLW21]). *Let $q > 2$ be any integer.*

- *Let $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ be two full rank matrices with $m > n$, $\mathbf{T}_\mathbf{A}$ be a trapdoor matrix for $\mathbf{A}$, a matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$ and $s \geq \|\mathbf{T}_\mathbf{A}\| \cdot \omega(\sqrt{\log m})$. Then there exists a PPT algorithm $\mathsf{SampleLeft}(\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{B}, \mathbf{U}, s)$ that outputs a sample from $[\mathbf{A}\|\mathbf{B}]^{-1}(\mathbf{U}, s)$.*
- *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ be a full rank matrix with $m > n$, $\mathbf{R} \in \mathbb{Z}^{m \times m}$, $\mathbf{U} \in \mathbb{Z}_q^{n \times \ell}$, $y \in \mathbb{Z}_q$ with $y \neq 0$, and $s \geq \sqrt{5} \cdot s_1(\mathbf{R}) \cdot \omega(\sqrt{\log m})$, where $s_1(\mathbf{R})$ denotes the spectral norm of $\mathbf{R}$. Then there exists a PPT algorithm $\mathsf{SampleRight}(\mathbf{A}, \mathbf{R}, y, \mathbf{U}, s)$ that outputs a sample from $[\mathbf{A}\|\mathbf{A} \cdot \mathbf{R} + y\mathbf{G}]^{-1}(\mathbf{U}, s)$.*
- *Let $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$, and two values $\rho, s \in \mathbb{R}$. Then there exists a PPT algorithm $\mathsf{Sampler\text{-}2}(\mathbf{R}, \rho, s)$ that runs in two stages. The output of the first stage is a full rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ for $m > n$, a vector $\mathbf{u} \in \mathbb{Z}_q^m$ along with its preimage $\mathbf{x} \in \mathbf{A}^{-1}(\mathbf{u}, \rho^2 + s^2)$. The output of the second stage is a sample from $[\mathbf{A}\|\mathbf{A}\mathbf{R}]^{-1}(\mathbf{u}, s)$.*

Learning with Errors. We recall the learning with errors assumption and its variants.

Assumption 1 (LWE). *Given $n, m, q, \chi \in \mathbb{N}$, the $\mathsf{LWE}_{n,m,q,\chi}$ assumption states that*

$$(\mathbf{A}, \mathbf{s}\mathbf{A} + \mathbf{e}) \approx_c (\mathbf{A}, \mathbf{c}),$$

where

$$\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}, \mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{e} \leftarrow \mathcal{D}_{\mathbb{Z},\chi}^m, \mathbf{c} \leftarrow \mathbb{Z}_q^m.$$

Lattice evaluation. We recall Tsabary's presentation [Tsa19] of the LWE evaluation procedure introduced in [BGG+14].

Lemma 7 (Fully Homomorphic Computation [BGG+14]). *Fix parameters n, q, ℓ and $m = \mathcal{O}(n \log q)$. There exists a pair of deterministic algorithms $(\mathsf{EvalF}, \mathsf{EvalFX})$ with the following properties.*

- *$\mathbf{A}_f \in \mathbb{Z}_q^{n \times m} \leftarrow \mathsf{EvalF}(\mathbf{A}, f)$. Here $\mathbf{A} \in \mathbb{Z}_q^{n \times m\ell}$ and $f : \{0,1\}^\ell \to \{0,1\}$ is a circuit.*
- *$\mathbf{H}_{\mathbf{A},f,\mathbf{u}} \in \mathbb{Z}_q^{m\ell \times m} \leftarrow \mathsf{EvalFX}(\mathbf{A}, f, \mathbf{u})$. Here $\mathbf{u} \in \{0,1\}^\ell$ is a binary string whose first bit is 0 and the second bit is 1, and $f : \{0,1\}^\ell \to \{0,1\}$ is a circuit with depth d that ignores the first and the second bit of the input.[22] Then we have:*

$$[\mathbf{A} - \mathbf{u} \otimes \mathbf{G}]\mathbf{H}_{\mathbf{A},f,\mathbf{u}} = \mathbf{A}_f - f(\mathbf{u})\mathbf{G} \mod q,$$

  where we denote $[u_1\mathbf{G}\| \cdots \|u_\ell\mathbf{G}]$ by $\mathbf{u} \otimes \mathbf{G}$. Furthermore, we have $\|\mathbf{H}_{\mathbf{A},f,\mathbf{u}}\|_\infty \leq m \cdot 2^{\mathcal{O}(d)}$. Finally, we have that the topmost m rows of $\mathbf{H}_{\mathbf{A},f,\mathbf{u}}$ constitutes an identity matrix.
- *The running time of $(\mathsf{EvalF}, \mathsf{EvalFX})$ is bounded by $\mathsf{poly}(n, m, \log q, 2^d)$.*

We also require the standard smudging lemma.

Lemma 8 (Smudging Lemma [WWW22]). *Take any $a \in \mathbb{Z}$ where $|a| \leq B$. Suppose $\chi \geq B\lambda^{\omega(1)}$. Then the statistical distance between the distributions $\{z : z \leftarrow \mathcal{D}_{\mathbb{Z},\chi}\}$ and $\{z + a : z \leftarrow \mathcal{D}_{\mathbb{Z},\chi}\}$ is $\mathsf{negl}(\lambda)$.*

[22] We refer the reader to [ARYY23, Remark 2.7] for an explanation of this requirement.

8.2 Our Construction

In the following construction, we assume public parameters

$$\mathsf{pp} := (\lambda, k, n_0, t, \ell, d, X, Y, p, q, n, m, Q, N, v),$$

consisting of the security parameter λ and the following quantities:

- The description of a tag space $\mathcal{T} = \{0,1\}^t$ for some $t = \mathsf{poly}(\lambda)$, such that tags may be arbitrary strings (e.g., time period numbers or dataset names).
- Let $k = \mathsf{poly}(\lambda)$ be the number of users, the function space is the set of all inner-product functions $f_{\mathbf{y}} : \mathbb{Z}_p^{n_0 \cdot k} \to Z_p$ indexed by an integer vector $\mathbf{y} \in \mathbb{Z}_p^{n_0 \cdot k}$. The message space will be $\mathcal{X} := \mathbb{Z}_p^{n_0 \cdot k}$. The vectors satisfy the following bounds: $\|\mathbf{x}_i\|_\infty < X, \|\mathbf{y}_i\|_\infty < Y$ for $i \in [k]$, and $p > n_0 \cdot k \cdot X \cdot Y$.
- Let ℓ be the length of the attribute in each slot. The construction supports general policy circuits $g : \{0,1\}^\ell \to \{0,1\}$ with bounded depth d and the decryption is possible when $g(\mathbf{u}) = 0$, where $\mathbf{u}$ is the attribute associated with each client.
- Let Q be the upper bound of 0-key (i.e., decryptable key) queries.

Let $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$, be the gadget matrix of rank n. For any $\mathsf{k} \in \{0,1\}^\lambda$, $\mathsf{PRF}(\mathsf{k}, \cdot) : \{0,1\}^t \to \mathbb{Z}_p^{n_0 \cdot k}$ is a pseudorandom function. Our construction goes as follows.

$\underline{(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(\mathsf{pp}, 1^k)}$**:** On input the public parameter pp and a number of users k, do the following:

- Sample $(\mathbf{A}, \mathbf{A}_\tau^{-1}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$ from Lemma 5 where $\mathbf{A}$ is uniform full-rank matrix in $\mathbb{Z}_q^{n \times m}$.
- Choose uniformly random matrices $\mathbf{B} \in \mathbb{Z}_q^{n \times m\ell}$, $\mathbf{P}_j \in \mathbb{Z}_q^{n \times n_0}$ for $j \in [N]$.
- For $i \in [k], j > i$: sample $\mathsf{k}_{i,j} = \mathsf{k}_{j,i} \xleftarrow{\$} \{0,1\}^\lambda$. Define $\mathsf{ek}_i := \left(\mathsf{k}_{i,j}\right)_{j \in [k]}$.
- Output

$$\mathsf{mpk} := \left(\mathsf{pp}, \mathbf{A}, \mathbf{B}, \{\mathbf{P}_j\}_{j \in [N]}\right); \quad \mathsf{msk} := \left(\mathbf{A}_\tau^{-1}, (\mathsf{ek}_i)_{i=1}^k\right)$$

$\underline{\mathsf{dk}_{\mathbf{y},g} \leftarrow \mathsf{Extract}(\mathsf{msk}, f_{\mathbf{y}}, g)}$**:** Given the master secret key msk, a function $f_{\mathbf{y}} : \mathbb{Z}_p^{n_0 \cdot k} \to \mathbb{Z}_p$ defined by an integer vector $\mathbf{y} := [\mathbf{y}_1 \| \cdots \| \mathbf{y}_k] \in \mathbb{Z}_p^{n_0 \cdot k}$ which maps an input $\mathbf{x} := [\mathbf{x}_1 \| \cdots \| \mathbf{x}_k] \in \mathbb{Z}_p^{n_0 \cdot k}$ to $f_{\mathbf{y}}(\mathbf{x}) := \langle \mathbf{x}, \mathbf{y} \rangle \in \mathbb{Z}_p$, and a policy function $g : \{0,1\}^\ell \to \{0,1\}$, do the following:

- Compute $\mathbf{B}_g \leftarrow \mathsf{EvalF}(\mathbf{B}, g)$
- Sample a random subset $\Delta \subset [N]$ with $|\Delta| = v$ according to the sampler $\mathsf{Sampler}_{\mathsf{Set}}(N, Q, v)$, and compute the subset sum $\mathbf{P}_\Delta \leftarrow \sum_{j \in \Delta} \mathbf{P}_j$.
- Sample matrix $\mathbf{J} \leftarrow \mathcal{D}_{\mathbb{Z},\rho}^{m \times n_0}$ and let $\mathbf{U} \leftarrow \mathbf{P}_\Delta - \mathbf{A}\mathbf{J} \mod q$.
- Sample $\mathbf{K}_g \leftarrow [\mathbf{A} \| \mathbf{B}_g]_\tau^{-1}(\mathbf{U}) + \begin{pmatrix} \mathbf{J} \\ \mathbf{0} \end{pmatrix}$ using $\mathbf{A}_\tau^{-1}$.
- Output the decryption key $\mathsf{dk}_{\mathbf{y},g} := (\Delta, \mathbf{R}_g \leftarrow \mathbf{K}_g \cdot \mathbf{y}^\top)$.

$\underline{\mathbf{c}_{i,\mathsf{tag},\mathbf{u}} \leftarrow \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, \mathsf{tag}, \mathbf{u})}$**:** Given mpk, the encryption key ek_i, a message $\mathbf{x}_i \in \mathbb{Z}_p^{n_0}$ with tag $\mathsf{tag} \in \{0,1\}^t$ and an attribute $\mathbf{u} \in \{0,1\}^\ell$, do the following:

- Compute $\mathbf{t}_{i,\mathsf{tag}} \leftarrow \sum_{j \neq i} (-1)^{j<i} \mathsf{PRF}(\mathsf{k}_{i,j}, \mathsf{tag} \| \mathbf{u}) \in \mathbb{Z}_p^{n_0 k}$
- Set $\mathbf{w}_{i,\mathsf{tag}} \leftarrow \left[0 \| \cdots \| 0 \| \mathbf{x}_i \| 0 \| \cdots \| 0\right] + \mathbf{t}_{i,\mathsf{tag}} \mod p$.
- Sample $\mathbf{s} \leftarrow \mathbb{Z}_q^n, \mathbf{R}_j \leftarrow \{-1,1\}^{m \times m}$ for $j \in [\ell]$, and error terms $\mathbf{e}_1 \leftarrow \mathcal{D}_{\mathbb{Z},\chi_1}^m, \mathbf{e}_{2,j} \leftarrow \mathcal{D}_{\mathbb{Z},\chi_2}^{n_0}$ for $j \in [N]$.
- Set an encoding of $\mathbf{w}_{i,\mathsf{tag}} \in \mathbb{Z}_p^{n_0 k}$ as $\{\mathbf{w}'_{i,\mathsf{tag},j}\}_{j \in [N]}$, where $\mathbf{w}'_{i,\mathsf{tag},j} = \frac{\lceil q/2 \rceil}{v} \mathbf{w}_{i,\mathsf{tag}}$.
- Let $\mathbf{e} \leftarrow \mathbf{e}_1 \cdot [\mathbf{I}_m \| \mathbf{R}_1 \| \cdots \| \mathbf{R}_\ell] \in \mathbb{Z}_q^{(\ell+1)m}$.
- Compute $\mathbf{c}_1 \leftarrow \mathbf{s}[\mathbf{A} \| \mathbf{B} - \mathbf{u} \otimes \mathbf{G}] + \mathbf{e}$.
- Compute $\mathbf{c}_{2,j} \leftarrow \mathbf{s}\mathbf{P}_i + \mathbf{e}_{2,j} + p^{e-1} \cdot \mathbf{w}'_{i,\mathsf{tag},j}$ for $j \in [N]$.
- Output the ciphertext $\mathbf{c}_{i,\mathsf{tag},\mathbf{u}}$ as $\mathsf{ct}_{i,\mathsf{tag},\mathbf{u}} := (\mathbf{c}_1, \{\mathbf{c}_{2,j}\}_{j \in [N]})$.

$\underline{\mathsf{Decrypt}(\mathsf{mpk}, \mathsf{dk}_{\mathbf{y},g}, \mathsf{tag}, \mathsf{ct})}$**:** On input a functional decryption key $\mathsf{dk}_{\mathbf{y},g}$ for a vector $\mathbf{y} \in \mathbb{Z}_p^{n_0 k}$, a tag $\mathsf{tag} \in \{0,1\}^t$ and a k-vector of ciphertexts $\mathsf{ct} := (\mathsf{ct}_{1,\mathsf{tag},\mathbf{u}}, \ldots, \mathsf{ct}_{k,\mathsf{tag},\mathbf{u}})$, do the following:

- If $g(\mathbf{u}) \neq 0$, output $\perp$.

- Compute $\mathbf{H}_{\mathbf{B},g,\mathbf{u}} \leftarrow \mathsf{EvalFX}(\mathbf{B}, g, \mathbf{u})$.
- For each $i \in [k]$, parse $\mathsf{ct}_{i,\mathsf{tag},\mathbf{u}}$ as $(\mathbf{c}_{1,i}, \{\mathbf{c}_{2,i,j}\}_{j\in[N]})$ then compute the following:

$$\mathbf{d}_{1,i} \leftarrow \mathbf{c}_{1,i} \cdot \begin{bmatrix} \mathbf{I}_m \\ \mathbf{H}_{\mathbf{B},g,\mathbf{u}} \end{bmatrix}, \qquad \mathbf{d}_{2,i} \leftarrow \sum_{j\in\Delta} \mathbf{c}_{2,i,j}, \qquad \mathbf{d}_{3,i} \leftarrow \mathbf{d}_{1,i} \cdot \mathbf{R}_g - \mathbf{d}_{2,i} \cdot \mathbf{y}^\top.$$

- Compute $\mu' \leftarrow \sum_i^k \mathbf{d}_{3,i}$ and output the value $\mu \in \mathbb{Z}_p$ that minimizes $\left|p^{e-1} \cdot \mu - \mu'\right|$.

8.3 Correctness

Correctness. Here, we show correctness of the scheme. Fix $g, \mathbf{u}$ such that $g(\mathbf{u}) = 0$.

We have:

$$\begin{aligned}
\mathbf{d}_{1,i} &= \mathbf{c}_{1,i} \cdot \begin{bmatrix} \mathbf{I}_m \\ \mathbf{H}_{\mathbf{B},g,\mathbf{u}} \end{bmatrix} = \mathbf{s}_i[\mathbf{A}\|\mathbf{B}_g] + \mathbf{e}'_{1,i} \\
\mathbf{d}_{2,i} &= \sum_{j\in\Delta} \mathbf{c}_{2,i,j} = \mathbf{s}_i\mathbf{P}_\Delta + \mathbf{e}'_{2,i} \\
\mathbf{d}_{3,i} &= \mathbf{d}_{1,i}\mathbf{R}_g - \mathbf{d}_{2,i}\mathbf{y}^\top = (\mathbf{e}'_{1,i} \cdot \mathbf{R}_g - \mathbf{e}'_{2,i}) \cdot \mathbf{y}^\top + p^{e-1}\langle \mathbf{w}_{i,\mathsf{tag}}, \mathbf{y}\rangle, \\
\mu' &= \sum_{i\in[k]} \mathbf{d}_{3,i} = \mathbf{e}_4 + p^{e-1}\langle \mathbf{x}, \mathbf{y}\rangle
\end{aligned}$$

where we define $\mathbf{e}'_{1,i} = \mathbf{e}_i \cdot \begin{bmatrix} \mathbf{I}_m \\ \mathbf{H}_{\mathbf{B},g,\mathbf{u}} \end{bmatrix}$, $\mathbf{e}'_{2,i} = \sum_{j\in\Delta} \mathbf{e}_{2,i,j}$, and $\mathbf{e}_4 = \sum_{i\in[k]}(\mathbf{e}'_{1,i} \cdot \mathbf{R}_g - \mathbf{e}'_{2,i}) \cdot \mathbf{y}^\top$. The last line follows from the fact that $(\mathbf{t}_{i,\mathsf{tag}})_{i\in[k]}$ are pseudorandom zero-shares. Thus, writing $\mathbf{w}_i \leftarrow [0\|\cdots\|0\|\mathbf{x}_i\|0\|\cdots\|0] + \mathbf{t}_{i,\mathsf{tag}} \mod p$, we have $\sum_{i=1}^k \mathbf{w}_{i,\mathsf{tag}} \mod p = \mathbf{x} \mod p \in \mathbb{Z}_p^{n_0 k}$.

The error term is bounded as follows.

$$\begin{aligned}
\|\mathbf{e}'_4\|_\infty &\le \max_{i\in[k]} k \cdot \left(\left\|(\mathbf{e}'_{1,i} \cdot \mathbf{R}_g - \mathbf{e}'_{2,i}) \cdot \mathbf{y}^\top\right\|_\infty\right) \\
&\le \max_{i\in[k]} k \cdot \left(\left\|\mathbf{e}'_{1,i} \cdot \mathbf{R}_g \cdot \mathbf{y}^\top\right\|_\infty + \left\|\mathbf{e}'_{2,i} \cdot \mathbf{y}^\top\right\|_\infty\right) \\
&\le k \cdot (\chi_1\beta\rho\tau Y + \chi_2 v n_0 Y)\,\mathsf{poly}(m) \le \beta_0.
\end{aligned}$$

Parameters. We set the parameters as follows.

$$\begin{aligned}
n &= \mathsf{poly}(\lambda, d, k, n_0), & m &= \mathcal{O}(n \log q), \\
\beta &= (2m)^d, & \rho &= \chi_1 = \chi_2 = \lambda^{\omega(1)}, \\
\tau &= \mathcal{O}\left(\sqrt{nm \log q}\right), & \beta_0 &= k\beta\rho\tau\chi_1 vnY\lambda^{\omega(1)}, \\
q &= \beta_0\lambda^{\omega(1)} = p^e.
\end{aligned}$$

8.4 Security

Theorem 8. *Under the* LWE *assumption, our construction given in Section 8.2 is IND-CPA secure with*

- *weak admissibility with one-challenge, complete, and repetitions on private inputs(Definition 3),*
- *adaptive corruption,*
- *selectively chosen attributes,*
- *adaptively chosen policies and functions.*

Proof (Sketch). The security of our construction relies on the compiled multi-client IPFE of [ALS16] using [ABG19]'s compiler. The proof follows identically as the original ABIPFE scheme in the single-input setting of [LLW21]. At a high level, the bulk of the proof is to simulate the key generation without msk, i.e., the trapdoor $\mathbf{A}^{-1}$. It was done by using SampleRight to answer 1-key queries, and use Sampler-2 from Lemma 5 to answer 0-key queries. We note that weak admissibility and adaptive corruption comes from the security of the underlying MCFE scheme compiled from [ALS16] using [ABG19]'s compiler. Selectively chosen attributes, and adaptively chosen policies and functions is due to [LLW21, Theorem 7.7], where the attributes need to be chosen selectively for the decryption key simulation. We refer the reader to [LLW21, Theorem 7.7] for which our proof is based on. □

Acknowledgments

We thank David Pointcheval for fruitful discussions on early stages of this work. This work was supported in part by the French ANR Project ANR-19-CE39-0011 PRESTO. Duong Hieu Phan was supported in part by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute. Xuan Thanh Do was supported by the 03/2025/KCM project (KCM/21-30 research program).

References

ABB10a. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 553–572. Springer, Berlin, Heidelberg, May / June 2010. 53

ABB10b. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 98–115. Springer, Berlin, Heidelberg, August 2010. 53

ABG19. Michel Abdalla, Fabrice Benhamouda, and Romain Gay. From single-input to multi-client inner-product functional encryption. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 552–582. Springer, Cham, December 2019. 4, 5, 7, 9, 10, 27, 29, 56

ABKW19. Michel Abdalla, Fabrice Benhamouda, Markulf Kohlweiss, and Hendrik Waldner. Decentralizing inner-product functional encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019, Part II*, volume 11443 of *LNCS*, pages 128–157. Springer, Cham, April 2019. 4, 7, 44

ABP+17. Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions: Extended abstract. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 2277–2293. ACM Press, October / November 2017. 3, 5

ACF+18. Michel Abdalla, Dario Catalano, Dario Fiore, Romain Gay, and Bogdan Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 597–627. Springer, Cham, August 2018. 44

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 467–497. Springer, Cham, December 2020. 4, 9

AGT21. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In Kobbi Nissim and Brent Waters, editors, *TCC 2021, Part II*, volume 13043 of *LNCS*, pages 224–255. Springer, Cham, November 2021. 7, 9

AKYY23. Shweta Agrawal, Simran Kumari, Anshu Yadav, and Shota Yamada. Broadcast, trace and revoke with optimal parameters from polynomial hardness. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 605–636. Springer, Cham, April 2023. 3

ALMT20. Shweta Agrawal, Benoît Libert, Monosij Maitra, and Radu Titiu. Adaptive simulation security for inner product functional encryption. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 34–64. Springer, Cham, May 2020. 11, 44

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part III*, volume 9816 of *LNCS*, pages 333–362. Springer, Berlin, Heidelberg, August 2016. 9, 11, 44, 56

ARYY23. Shweta Agrawal, Mélissa Rossi, Anshu Yadav, and Shota Yamada. Constant input attribute based (and predicate) encryption from evasive and tensor LWE. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 532–564. Springer, Cham, August 2023. 53

ATY23. Shweta Agrawal, Junichi Tomida, and Anshu Yadav. Attribute-based multi-input FE (and more) for attribute-weighted sums. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 464–497. Springer, Cham, August 2023. 4

BF99. Dan Boneh and Matthew K. Franklin. An efficient public key traitor tracing scheme. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 338–353. Springer, Berlin, Heidelberg, August 1999. 3

BGG+14. Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 533–556. Springer, Berlin, Heidelberg, May 2014. 9, 12, 53

BGM+16. Andrej Bogdanov, Siyao Guo, Daniel Masny, Silas Richelson, and Alon Rosen. On the hardness of learning with rounding over small modulus. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 209–224. Springer, Berlin, Heidelberg, January 2016. 11

BLM+24. Pedro Branco, Russell W. F. Lai, Monosij Maitra, Giulio Malavolta, Ahmadreza Rahimi, and Ivy K. Y. Woo. Traitor tracing without trusted authority from registered functional encryption. In Kai-Min Chung and Yu Sasaki, editors, *ASIACRYPT 2024, Part III*, volume 15486 of *LNCS*, pages 33–66. Springer, Singapore, December 2024. 3

BLP+13. Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th ACM STOC*, pages 575–584. ACM Press, June 2013. 53

BN08. Dan Boneh and Moni Naor. Traitor tracing with constant size ciphertext. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 501–510. ACM Press, October 2008. 3

BP08. Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In Reihaneh Safavi-Naini, editor, *ICITS 08*, volume 5155 of *LNCS*, pages 171–182. Springer, Berlin, Heidelberg, August 2008. 3

BPR12. Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 719–737. Springer, Berlin, Heidelberg, April 2012. 11

BPR24. Dan Boneh, Aditi Partap, and Lior Rotem. Accountability for misbehavior in threshold decryption via threshold traitor tracing. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VII*, volume 14926 of *LNCS*, pages 317–351. Springer, Cham, August 2024. 3

BS95. Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data (extended abstract). In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 452–465. Springer, Berlin, Heidelberg, August 1995. 3

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 573–592. Springer, Berlin, Heidelberg, May / June 2006. 3, 23

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Berlin, Heidelberg, March 2011. 3

BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 211–220. ACM Press, October / November 2006. 3

BZ14. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 480–499. Springer, Berlin, Heidelberg, August 2014. 3

CDG+18a. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Decentralized multi-client functional encryption for inner product. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 703–732. Springer, Cham, December 2018. 3, 4, 5, 7, 8, 9, 13

CDG+18b. Jérémy Chotard, Edouard Dufour Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021, 2018. 4, 7, 14

CDSG+20. Jérémy Chotard, Edouard Dufour-Sans, Romain Gay, Duong Hieu Phan, and David Pointcheval. Dynamic decentralized functional encryption. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 747–775. Springer, Cham, August 2020. 4, 7, 13, 14

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 257–270. Springer, Berlin, Heidelberg, August 1994. 3

CHKP10. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 523–552. Springer, Berlin, Heidelberg, May / June 2010. 53

CVW+18. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from LWE made simple and attribute-based. In Amos Beimel and Stefan Dziembowski, editors, *TCC 2018, Part II*, volume 11240 of *LNCS*, pages 341–369. Springer, Cham, November 2018. 3, 47

DPP20. Xuan Thanh Do, Duong Hieu Phan, and David Pointcheval. Traceable inner product functional encryption. In Stanislaw Jarecki, editor, *CT-RSA 2020*, volume 12006 of *LNCS*, pages 564–585. Springer, Cham, February 2020. 3, 5, 6, 7

DPY20. Xuan Thanh Do, Duong Hieu Phan, and Moti Yung. A concise bounded anonymous broadcast yielding combinatorial trace-and-revoke schemes. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20International Conference on Applied Cryptography and Network Security, Part II*, volume 12147 of *LNCS*, pages 145–164. Springer, Cham, October 2020. 3

GGG+14. Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 578–602. Springer, Berlin, Heidelberg, May 2014. 3, 4, 5

GKL+13. S. Dov Gordon, Jonathan Katz, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. Cryptology ePrint Archive, Report 2013/774, 2013. 3, 5

GKRW18. Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 467–497. Springer, Cham, August 2018. 37

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 660–670. ACM Press, June 2018. 3, 15, 23, 37, 47

GKW19. Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 149–179. Springer, Cham, December 2019. 3, 11, 23, 26, 27, 34, 35, 37, 38, 39, 40, 47

GLW23. Junqing Gong, Ji Luo, and Hoeteck Wee. Traitor tracing with $N^{1/3}$-size ciphertexts and $O(1)$-size keys from k-Lin. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part III*, volume 14006 of *LNCS*, pages 637–668. Springer, Cham, April 2023. 3

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th ACM STOC*, pages 197–206. ACM Press, May 2008. 53

KW20. Sam Kim and David J. Wu. Collusion resistant trace-and-revoke for arbitrary identities from standard assumptions. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 66–97. Springer, Cham, December 2020. 3

LAKWH22. Fucai Luo, Saif Al-Kuwari, Haiyan Wang, and Weihong Han. Generic construction of trace-and-revoke inner product functional encryption. In Vijayalakshmi Atluri, Roberto Di Pietro, Christian Damsgaard Jensen, and Weizhi Meng, editors, *ESORICS 2022, Part I*, volume 13554 of *LNCS*, pages 259–282. Springer, Cham, September 2022. 3, 5, 6, 7, 27

LLW21. Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption - stronger security and smaller ciphertexts. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 498–527. Springer, Cham, October 2021. 9, 10, 53, 56

LPSS14. San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-LWE and applications in traitor tracing. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 315–334. Springer, Berlin, Heidelberg, August 2014. 3

LT19. Benoît Libert and Radu Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 520–551. Springer, Cham, December 2019. 4, 5, 7, 44

LYJP14. Benoît Libert, Moti Yung, Marc Joye, and Thomas Peters. Traceable group encryption. In Hugo Krawczyk, editor, *PKC 2014*, volume 8383 of *LNCS*, pages 592–610. Springer, Berlin, Heidelberg, March 2014. 3

MP12. Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 700–718. Springer, Berlin, Heidelberg, April 2012. 53

Ngu24. Duy Nguyen. Dynamic decentralized functional encryptions from pairings in the standard model. Cryptology ePrint Archive, Paper 2024/580, 2024. https://eprint.iacr.org/2024/580. 4

NPP22. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with fine-grained access control. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part I*, volume 13791 of *LNCS*, pages 95–125. Springer, Cham, December 2022. 9, 14

NPP23. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Optimal security notion for decentralized multi-client functional encryption. In Mehdi Tibouchi and Xiaofeng Wang, editors, *ACNS 23International Conference on Applied Cryptography and Network Security, Part II*, volume 13906 of *LNCS*, pages 336–365. Springer, Cham, June 2023. 4, 5, 7, 8, 9, 13, 14, 26

NPP25. Ky Nguyen, Duong Hieu Phan, and David Pointcheval. Multi-client functional encryption with public inputs and strong security. In *Public-Key Cryptography - IACR PKC 2025*. Springer, 2025. 4, 7, 9, 13, 39

NPS24. Ky Nguyen, David Pointcheval, and Robert Schädlich. Decentralized multi-client functional encryption with strong security. *CiC*, 1(2):3, 2024. 4

NPS25. Ky Nguyen, David Pointcheval, and Robert Schädlich. Dynamic decentralized functional encryption: Generic constructions with strong security. In *Public-Key Cryptography - IACR PKC 2025*. Springer, 2025. 4, 9, 14

NWZ16. Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: How to embed arbitrary information in a key. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 388–419. Springer, Berlin, Heidelberg, May 2016. 3, 6

PS00. David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000. 17

SV23. Elaine Shi and Nikhil Vanjani. Multi-client inner product encryption: Function-hiding instantiations without random oracles. In Alexandra Boldyreva and Vladimir Kolesnikov, editors, *PKC 2023, Part I*, volume 13940 of *LNCS*, pages 622–651. Springer, Cham, May 2023. 4

Tsa19. Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 62–85. Springer, Cham, August 2019. 53

WWW22. Brent Waters, Hoeteck Wee, and David J. Wu. Multi-authority ABE from lattices without random oracles. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022, Part I*, volume 13747 of *LNCS*, pages 651–679. Springer, Cham, November 2022. 53

Zha20. Mark Zhandry. New techniques for traitor tracing: Size $N^{1/3}$ and more from pairings. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 652–682. Springer, Cham, August 2020. 3

Zha21. Mark Zhandry. White box traitor tracing. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 303–333, Virtual Event, August 2021. Springer, Cham. 8

ZZ23. Qiuwei Zheng and Jun Zhao. Trace-and-revoke quadratic functional encryption. In *Information Security*. Springer Nature Switzerland, 2023. 7