Multi-Client Inner Product Encryption: Function-Hiding Instantiations Without Random Oracles

Elaine Shi and Nikhil Vanjani*

Carnegie Mellon University

Abstract. In a Multi-Client Functional Encryption (MCFE) scheme, n clients each obtain a secret encryption key from a trusted authority. During each time step t, each client i can encrypt its data using its secret key. The authority can use its master secret key to compute a functional key given a function f, and the functional key can be applied to a collection of n clients' ciphertexts encrypted to the same time step, resulting in the outcome of f on the clients' data. In this paper, we focus on MCFE for *inner-product* computations.
If an MCFE scheme hides not only the clients' data, but also the function f, we say it is *function hiding*. Although MCFE for inner-product computation has been extensively studied, how to achieve function privacy is still poorly understood. The very recent work of Agrawal et al. showed how to construct a *function-hiding* MCFE scheme for inner-product assuming standard bilinear group assumptions; however, they assume the existence of a random oracle and prove only a relaxed, selective security notion. An intriguing open question is whether we can achieve function-hiding MCFE for inner-product *without* random oracles.
In this work, we are the first to show a function-hiding MCFE scheme for inner products, relying on standard bilinear group assumptions. Further, we prove *adaptive* security without the use of a random oracle. Our scheme also achieves succinct ciphertexts, that is, each coordinate in the plaintext vector encrypts to only $O(1)$ group elements.
Our main technical contribution is a new upgrade from single-input functional encryption for inner-products to a multi-client one. Our upgrade preserves function privacy, that is, if the original single-input scheme is function-hiding, so is the resulting multi-client construction. Further, this new upgrade allows us to obtain a conceptually simple construction.

Keywords: multi-client functional encryption, adaptive security, bilinear group

1 Introduction

Multi-Input Functional Encryption (MIFE), first proposed by Goldwasser et al. [19], allows us to evaluate certain functions on multiple users' encrypted

* Author ordering is randomized.

data. In MIFE, a trusted setup gives an encryption key to each of n users, and then each user i can use its encryption key to encrypt some value x_i. A data analyst can ask the trusted setup for a cryptographic token to evaluate a specific function f. Equipped with the token, the data analyst can evaluate the outcome $f(x_1, \ldots, x_n)$ when presented with n ciphertexts each encoding $x_1, \ldots, x_n$, respectively.

It is also well-understood that the MIFE formulation suffers from some limitations. For instance, it does not make any attempt to limit the mix-and-match of ciphertexts. The evaluator can take any combination of users' ciphertexts, one from each user, to evaluate the function f. As a simple example, imagine that two users each encrypted two values, x_0, x_1 and y_0, y_1, respectively. Then, the evaluator can learn the outcome of $f(x_{b_0}, y_{b_1})$ for any combination of $b_0, b_1 \in \{0, 1\}$. In some applications, this may be too much leakage, and we want to limit the extent of mix-and-match. As a result, a related notion called Multi-Client Functional Encryption (MCFE) was introduced [19, 26]. One way to understand the MCFE abstraction is to think of a "streaming" setting [26]: imagine that in every time step t, each user i encrypts a value $x_{i,t}$. Given the ciphertexts, the evaluator can evaluate $f(x_{1,t}, \ldots, x_{n,t})$ for each time step t, but it cannot mix-and-match the ciphertexts across different time steps and combine them in the evaluation. This greatly restricts the inherent leakage of the scheme. More generally, MCFE schemes allow users to encrypt to a label t, and only ciphertexts encrypted to the same label t can be used together during the functional evaluation. MCFE has numerous applications. For example, it has been applied to privacy-preserving, time-series data aggregation [26]. It is also useful in federated learning [14, 24] where a server may wish to (incrementally) train some machine learning model based on data collected from users' mobile devices for each period of time. Very recent work also showed that function-hiding MCFE schemes can be used to construct a non-interactive anonymous routing scheme [27].

In vanilla MCFE schemes, our goal is to hide the plaintexts. However, in some applications [10, 27], we also want an additional privacy property: not only should the ciphertexts hide the underlying messages, we also want the tokens to hide the function f being evaluated. An MCFE scheme that achieves this extra property is said to be *function-hiding* or *function-private* [10].

Status quo of our knowledge. The holy grail is to be able to construct MCFE for general functions from standard assumptions. However, it is believed that supporting general functions may be no easier than achieving indistinguishability obfuscation [11, 13, 21]. On the other hand, assuming the existence of indistinguishability obfuscation and the existence of a random oracle, we can indeed construct (function-revealing) MCFE for general functions [19].

Given that indistinguishability obfuscation will unlikely become practical in the near term, a natural question is for which functions can we construct efficient MCFE schemes, and ideally from standard assumptions? Along this direction, a line of work has explored how to construct (function-revealing) MCFE schemes for inner product computation, also called Multi-Client Inner-Product

Encryption (MCIPE)[1]. This exploration culminated in the work of Libert and Titiu [22], who showed how to construct an adaptively secure, function-revealing MCFE from standard lattice assumptions; and moreover, their scheme achieves succinct ciphertexts (i.e., each client's ciphertext size does not grow w.r.t. the number of parties). An independent work of Abdalla et al. [1] also achieved almost the same result as Libert and Titiu [22], except that 1) they can instantiate their constructions from DDH, LWE, or DCR assumptions; and 2) their ciphertexts are not succinct and grow linearly in the number of clients. Besides the work of Libert and Titiu [22] and that of Abdalla et al. [1], all other MCIPE constructions, even in the function-revealing setting, rely on random oracles for proving security [4, 15, 16].

When it comes to function privacy, however, our knowledge is relatively little. So far, the only known function-hiding MCIPE construction is the elegant work by Agrawal et al. [10], who constructed such a scheme from standard bilinear group assumptions, and additionally, assuming the existence of a random oracle; moreover, their construction is only *selectively* secure. To date, it remains elusive how to construct a function-hiding MCIPE scheme without random oracles.

Therefore, the status quo of MCIPE begs the following natural questions:

1. Can we construct an MCIPE scheme with *succinct ciphertexts* from *non-lattice* assumptions and *without random oracles*? This question is open *even for selective security and without requiring function privacy.*
2. Can we construct a *function-hiding* MCIPE scheme from *any standard* assumptions, *without random oracles*? This question is open *even for selective security, and even without caring about efficiency.*

Recall that the recent lower bound result Ünal [29] suggests that one cannot hope to achieve function-private (even single-input) inner-product encryption from lattices using a class of natural approaches. Therefore, being able to answer the first question above could open up more avenues towards eventually getting function privacy (i.e., the second question).

1.1 Our Results and Contributions

In this paper, we present a new MCIPE scheme from standard bilinear groups assumptions (against polynomial-time adversaries), and we prove the scheme to satisfy adaptive, function-hiding security. Our scheme is concretely efficient in the sense that every coordinate in the plaintext vector encrypts to only $O(1)$ group elements, and every coordinate in a key vector will result in $O(1)$ group elements in the functional key.

Therefore, we not only provide an affirmative answer to the above open questions, we also achieve all the desirable properties in a single unifying construction. More specifically, we prove the following theorem.

[1] Throughout this paper, the term "inner-product encryption" always means "inner-product *functional* encryption". This terminology is standard in this space.

Table 1: Comparison with prior MCIPE schemes where $O_\lambda(\cdot)$ hides terms related to the security parameter λ.

Scheme	Assumptions	Func privacy	Adaptive	per-coordinate CT
[15]	DDH + RO	✘	✔	$O_\lambda(1)$
[1]	DDH or DCR or LWE	✘	✔	$O_\lambda(n)$
[22]	LWE	✘	✔	$O_\lambda(1)$
[4]	(bilinear or DCR or LWE) + RO	✘	✔	$O_\lambda(1)$
[10]	bilinear + RO	✔	✘	$O_\lambda(1)$
Our work	bilinear	✔	✔	$O_\lambda(1)$

Theorem 1. *Suppose that the Decisional Linear (DLin) and Decisional Bilinear Diffie-Hellman (DBDH) assumptions hold in suitable bilinear groups. There exists an MCIPE scheme that satisfies adaptive, function-hiding, indistinguishability-based security. Moreover, the scheme achieves succinct ciphertext.*

Techniques: a function-privacy-preserving upgrade from single-input to multi-client. Notably, our MCIPE construction is conceptually simpler than some prior (even function-revealing) constructions. Since the *conceptual simplicity* could make it easier for future work to further extend and improve our framework, we believe it yet another contribution made by our work.

To get our result, we describe a new upgrade from a single-input inner-product encryption (IPE) to MCIPE. Further, if the underlying IPE scheme satisfies adaptive function-hiding security, so does the resulting MCIPE scheme. We believe our upgrade technique can be of independent interest. Previously, a couple works [1,10] also take the approach of upgrading from a single-input IPE scheme; however, previous techniques suffer from several drawbacks. Abdalla et al. [1] showed how to upgrade a single-input IPE scheme to a multi-client one. Their technique suffers from a couple drawbacks: 1) even if the original IPE scheme is function-private, their upgrade does not preserve function privacy; and 2) their upgrade incurs a $\Theta(n)$ blowup in the per-client ciphertext size. The recent work of Agrawal et al. [10] can also be viewed as an upgrade from a function-hiding IPE to a function-hiding MCIPE scheme — however, as mentioned, their construction critically relies on a random oracle and is only selectively secure.

We compare our contributions with prior work in Table 1 where n denotes the total number of clients.

1.2 Additional Related Work

We now review related work, and explain why some of those ideas do not easily extend to our new result.

Multi-input functional encryption. As mentioned, multi-input functional encryption (MIFE), originally proposed by Goldwasser et al. [19], can be viewed as a weakening of MCFE where all ciphertexts are encrypted to the same label. This relaxation often makes constructing MIFE easier. For example, for general functions, we know how to construct MIFE assuming indistinguishability obfuscation and other standard cryptographic assumptions. However, when it comes to MCFE for general functions, we not only need indistinguishability obfuscation but also the random oracle model (unless we can publish separate public parameters for each different label that will ever be encountered).

A line of work explored how to construct MIFE for inner-product from standard assumptions. This line culminated in the work of Abdalla et al. [5], who showed a construction that satisfies adaptive function-hiding security, assuming standard bilinear group assumptions, and achieving succinct ciphertexts. Again, their technique does not easily give rise to a multi-*client* counterpart. In fact, without the use of a random oracle, we do not even know how to construct a *function-revealing* non-lattice-based MCIPE scheme with succinct ciphertexts, let alone a function-hiding one; and for getting function privacy, it is believed that there may be potential barriers using lattice techniques [29].

Recently, Agrawal et al. [9] showed how to construct MIFE for *quadratic functions* — however, their scheme does not allow corruption of a subset of the clients and therefore does not directly extend to the multi-client setting; moreover, their scheme is not function hiding. Abdalla et al. [7] showed how to construct a 2-round MCFE scheme for *quadratic functions*. In their construction, encryption involves a 2-round interaction between a client and a set of authorities. Moreover, their scheme is not function hiding.

Throughout our paper (including Table 1), we assume static corruption. Besides our notion of adaptive security where encryption and key queries can be chosen adaptively by the adversary, Abdalla et al. [1,2], Libert and Titiu [22] and Nguyen et al. [25] also considered a different, adaptive corruption notion, where the clients are corrupted in an adaptive fashion — however, their constructions are non-function-hiding. Abdalla et al. [2] constructed an MIFE scheme for adaptive corruptions but the scheme is not function-hiding. Abdalla et al. [1] and Libert and Titiu [22] obtained similar results in the stronger multi-client setting from pairings and lattice assumptions respectively. Nguyen et al. [25] constructed MCFE with fine-grained access control from pairings in the RO model. Our work can also secure against adaptive corruption if we make subexponential assumptions and use standard complexity leveraging techniques. To the best of our knowledge, *no known technique can achieve adaptive corruption for the function-hiding setting* without relying on complexity leveraging, even for multi-input inner-product encryption, and even for selective-query security. How to achieve security against adaptive corruption in the function-private setting is an open question. Our current proof techniques adopt a sequence of hybrids that are incompatible with adaptive corruption — this also applies to other known constructions with function privacy [10, 27]: since we must answer the queries differently for corrupt coordinates and honest coordinates, the current proof framework will not work if the challenger does not have know this upfront.

The main challenge for adaptive corruption is that in our hybrid sequence, we make use of a multiple-slot trick tailored for the function-hiding setting — for example, switching from $(\mathbf{x}_i^{(1)}, 0^m, \ldots)$ and $(\mathbf{y}_i^{*(1)}, 0^m, \ldots)$ to $(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)})$ and $(\mathbf{y}_i^{*(1)}, 0^m, \ldots)$ for an honest coordinate i (see Hybrid Real_1 to Hyb_0 transition in Table 2). For adaptive corruption, if i is not corrupt yet but will eventually become corrupt, we should not make this switch for coordinate i — but we cannot predict whether i will be eventually corrupt. In the function-revealing schemes [1, 2, 22, 25], their proofs do not rely on this type of multiple-slot trick making it much easier to prove adaptive corruption.

Comparison with Shi and Wu [27]. Recently, the work of Shi and Wu [27] considered a simple special case of inner-product, that is, "selection". Selection is the task of selecting one coordinate from the plaintext vector, i.e., inner product with a special vector where one coordinate is set to 1, and all other coordinates are set to 0. They showed how to achieve a *selective*, function-hiding MCFE scheme for selection. Shi and Wu's framework cannot be easily extended to get our result. First, their proof technique only works for proving *selective* security, whereas we want to prove *adaptive* security. Second, their framework is tailored for selection and does not easily extend to general inner product computation. Specifically, to construct a *function-hiding* MCFE scheme for selection, they first construct a *function-revealing* MCFE scheme for selection without RO, and then perform a function-privacy upgrade. We are not able to follow the same paradigm, since Previously, it was not even known how to construct a non-lattice-based, *function-revealing* MCIPE scheme without RO and with *succinct ciphertexts*. The only known non-lattice-based, function-revealing MCIPE scheme without RO is the elegant work by Abdalla et al. [1]. Unfortunately, their scheme has an $\Theta(n)$ blowup in the ciphertext size that we want to avoid. Although it is known how to construct a *function-revealing* MCIPE scheme without RO using lattices [22], the recent lower bound result Ünal [29] suggests that one cannot hope to achieve function-private IPE from lattices using a class of natural approaches.

Decentralizing MIFE and MCFE schemes. An elegant line of work [8, 10, 15, 17] considers how to decentralize the key generation in multi-input and multi-client functional encryption schemes. The resulting schemes are typically referred to as ad-hoc MIFE [8] or as dynamic decentralized functional encryption (DDFE) [10, 17]. Roughly speaking, ad-hoc MIFE can be viewed as a generalization of MIFE, and DDFE can be viewed as a generalization of MCFE, where the key generation can be performed in a decentralized fashion without relying on a trusted party. This line of work culminated in the recent work of Agrawal et al. [10] who constructed a function-hiding DDFE scheme from bilinear groups in the random oracle model. Therefore, an interesting question left open by our work is whether there exists a function-hiding DDFE scheme from standard assumptions, without relying on a random oracle. This question is open even for selective security and even without caring about efficiency.

2 Overview of Our Constructions and Techniques

We now give an informal overview of our construction and proof techniques. In our subsequent technical sections, we will present formal definitions, detailed scheme description, and formal proofs.

Notations. Throughout, we will use boldface letters such as $\mathbf{x}$ to denote vectors. Given a bilinear group $\mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$ of prime order q, we use the notation $[\![x]\!]$ and $[\![x]\!]_T$ to denote the group encoding of $x \in \mathbb{Z}_q$ in the source and target groups; and a similar notation is used for vectors too.

2.1 Why Prior Work Needed a Random Oracle

The recent work of Agrawal et al. [10] suggested the following elegant idea for constructing a function-hiding MCFE scheme for inner-product (also called MCIPE). Let IPE be a *function-hiding* inner-product encryption scheme (i.e., a single-input FE scheme for inner-product). We assume that IPE is built from suitable bilinear groups. We additionally assume the following nice property about IPE: the encryption algorithm (denoted **Enc**) and the functional key generation algorithm (denoted **KGen**) should work even when taking in the group encoding of the plaintext or key vector rather than the vector itself.

Let $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ denote the plaintext vector where $\mathbf{x}_i$ is the component corresponding to client $i \in [n]$. let $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$ be the key vector where $\mathbf{y}$ is the component corresponding to client $i \in [n]$. Agrawal et al. [10]'s construction works as follows. Henceforth let $H(\cdot)$ be a random oracle and let $[\![\rho_t]\!] = H(t)$ which is a hash of the time step t (also called label).

$$\begin{array}{rcl} \underline{\text{Ciphertext:}} & & \underline{\text{Functional key:}} \\ \mathsf{ct}_1 = \mathsf{IPE}.\mathbf{Enc}(\mathsf{imsk}_1, [\![\mathbf{x}_1, \rho_t]\!]) & \Leftrightarrow & \mathsf{isk}_1 = \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_1, [\![\mathbf{y}_1, z_1]\!]) \\ \vdots & & \vdots \\ \mathsf{ct}_n = \mathsf{IPE}.\mathbf{Enc}(\mathsf{imsk}_n, [\![\mathbf{x}_n, \rho_t]\!]) & \Leftrightarrow & \mathsf{isk}_n = \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_n, [\![\mathbf{y}_n, z_n]\!]) \end{array}$$

In the above, each client $i \in [n]$ has an independent IPE instance whose master secret key is imsk_i also chosen by the trusted setup; the terms $z_1, \ldots, z_n$ are chosen freshly at random for each client respectively during each **KGen** query, such that their summation is 0, that is, $z_1 + z_2 + \ldots + z_n = 0$.

Henceforth, suppose $H(t) = [\![\rho_t]\!]$. To decrypt, we can $\mathsf{IPE}.\mathbf{Dec}(\mathsf{ct}_i, \mathsf{isk}_i)$ to obtain the partial decryption $\langle \mathbf{x}_i, \mathbf{y}_i\rangle + \rho_t \cdot z_i$ encoded as the exponent of some group element. When we sum up all the partial decryptions, the part $\rho_t \cdot z_1 + \rho_t \cdot z_2 + \ldots + \rho_t \cdot z_n$ cancel out, and we are left with $\langle \mathbf{x}, \mathbf{y}\rangle$.

Intuitively, the ciphertext terms $H(t)$ and key terms z_i's serve to re-randomize each partial decryption. In this way, the adversary is forced to use all n clients' ciphertext components from the same time step to yield a meaningful decryption result. If the adversary mixes and matches ciphertext components from different time steps, decryption gives garbage and no information is leaked. If the adversary uses a proper subset of the clients' ciphertext components but not all n of them, decryption also gives garbage. Agrawal et al. [10]'s scheme critically relies on a random oracle $H(\cdot)$ due to a combination of following reasons:

1. For functionality, the multiple clients must coordinate and put in a *common term* that is multiplied with the z_i's during the decryption. Only in this way, can the randomizing terms cancel out when all partial decryptions are summed up;
2. For security, the aforementioned common term must be *random*, and not only so, must be fresh for each time step t. Henceforth let $\mathcal{H}$ denote the set of honest clients. Without going into full details about their proof, basically, in some critical step in their hybrid sequence, they want to argue the following computational indistinguishability statement for some "challenge key" which involves the terms $z_1^*, \ldots, z_n^*$:

$$\{[\![\rho_t \cdot z_i^*]\!]\}_{i\in\mathcal{H},t=1,2,3,\ldots} \overset{c}{\equiv} \{R_{i,t}\}_{i\in\mathcal{H},t=1,2,3,\ldots} \quad (1)$$

   where $\{R_{i,t}\}_{i\in\mathcal{H},t=1,2,3,\ldots}$ are randomly chosen group elements such that the product is preserved in every time step, that is:

$$\forall t : \prod_{i\in\mathcal{H}} R_{i,t} = \prod_{i\in\mathcal{H}} [\![\rho_t \cdot z_i^*]\!] \quad (2)$$

   Agrawal et al. [10] argue that the above holds under the SXDH assumption as long as each $H(t) = [\![\rho_t]\!]$ is a random group element.

In summary, in the scheme by Agrawal et al., the random oracle $H(\cdot)$ allows the clients to coordinate without communication, and adopt the same random term that is refreshed for each t in their respective ciphertexts. One naïve way to avoid the RO is for the trusted step to publish all random $\{[\![\rho_t]\!]\}_{t=1,2,3\ldots}$ terms in the sky upfront, but then the scheme would not be able to support an unbounded number of time steps.

2.2 Removing the RO: A Strawman Idea

In Agrawal et al.'s scheme, the coordinated randomness $z_1, \ldots, z_n$ is part of the functional key; therefore, in the ciphertext, all clients must put in shared common randomness to pair with these terms. To remove the RO, a strawman idea is move the coordinated randomness to the ciphertext. To this end, we will employ a *correlated pseudorandom function*, denoted CPRF. In a CPRF scheme, each client $i \in [n]$ obtains a secret key K_i from a trusted setup. Then, given a message t, the user can compute $\mathsf{CPRF.Eval}(K_i, t)$ to obtain an outcome that is computationally indistinguishable from random, subject to the constraint that

$$\sum_{i\in[n]} \mathsf{CPRF.Eval}(K_i, t) = 0 \quad (3)$$

Further, even when a subset of the clients may be corrupted, the outcomes of the honest clients' evaluations are nonetheless pseudorandom subject to the constraint in Equation (3) — see Section 4.2 for the formal definition. Earlier works have shown how to construct such a CPRF assuming the existence of pseudorandom functions [1, 14]. With such a CPRF, we can construct the following strawman scheme where we use the shorthand notation $\mathsf{CPRF}(K_i, t) = \mathsf{CPRF}.\mathbf{Eval}(K_i, t)$, and z denotes a term shared across the different clients $1, \dots, n$ for the same functional key, but *freshly chosen for every functional key*:

$$\begin{array}{ccc}
\underline{\text{Ciphertext:}} & & \underline{\text{Functional key:}} \\
\mathsf{ct}_1 = \mathsf{IPE}.\mathbf{Enc}(\mathsf{imsk}_1, (\mathbf{x}_1, \mathsf{CPRF}(K_1, t)) & \Leftrightarrow & \mathsf{isk}_1 = \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_1, (\mathbf{y}_1, z)) \\
\vdots & & \vdots \\
\mathsf{ct}_n = \mathsf{IPE}.\mathbf{Enc}(\mathsf{imsk}_n, (\mathbf{x}_n, \mathsf{CPRF}(K_n, t)) & \Leftrightarrow & \mathsf{isk}_n = \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_n, (\mathbf{y}_n, z))
\end{array}$$

The use of the CPRF in the above allows the distributed clients to adopt correlated randomness that is refreshed for each t at encryption time, and thus avoids the RO. However, the strawman scheme does not work since the security proof fails to go through. Let $\mathcal{H}$ denote the set of honest clients. In a critical step Agrawal et al.'s proof, they rely on the security of the IPE scheme to hide the $\{z_i^*\}_{i\in\mathcal{H}}$ terms in some "challenge key", and instead move information about $[\![\rho_t \cdot z_i^*]\!]_{i\in\mathcal{H},t=1,2,3,\dots}$ into the ciphertext components — recall that in their scheme, the term $\rho_t \cdot z_i^*$ is the randomizing term that protects client i's message in the i-th partial decryption. They argue that the terms $[\![\rho_t \cdot z_i^*]\!]_{i\in\mathcal{H},t=1,2,3,\dots}$ are computationally indistinguishable from random except their product is conserved for every time step t — see Equations (1) and (2).

Unfortunately, this strategy no longer works, as now all the key components share the same randomness z. When the adversary corrupts a subset of the clients, it will learn info about the randomness $[\![z^*]\!]$ in the challenge key. Additionally, the adversary can gain info about $[\![\mathsf{CPRF}(K_i, t)]\!]_{i\in\mathcal{H},t=1,2,3,\dots}$ from the ciphertexts. Hence, the adversary can easily distinguish $\{[\![\mathsf{CPRF}(K_i, t) \cdot z^*]\!]\}_{i\in\mathcal{H},t=1,2,3,\dots}$ from random terms through a DDH-style attack. We stress that using asymmetric group and SXDH does not fix this attack, as the ciphertext and the key must come from opposite source groups to be paired with each other[2].

2.3 Our Selectively Secure Construction

We start with the goal of achieving selective security (i.e., assuming that the adversary must submit all **KGen** queries ahead of encryption queries), and later describe additional techniques for achieving adaptive security. We sketch our selectively secure construction below — a more formal presentation can be found in

[2] In Appendix E of the online full version, we show that a variant of the strawman scheme can indeed be proven secure in a different selective model, i.e., the adversary must submit all encryption queries ahead of **KGen** queries. However, we do not know any easy way to build from this selective scheme and get adaptive security eventually.

subsequent technical sections. Recall that IPE denotes a function-hiding (single-input) inner-product encryption scheme. In our scheme the public parameters are just the public parameters of the underlying function-hiding secure IPE scheme.

- **Setup**: we run n independent instances of $\mathsf{IPE}.\mathbf{Setup}$ to sample n secret keys denoted $\mathsf{imsk}_1, \ldots, \mathsf{imsk}_n$, respectively. We also run the setup algorithm of the CPRF, and obtain $K_1, \ldots, K_n$. Finally, we generate a random $a_i \xleftarrow{\$} \mathbb{Z}_q$ for each client $i \in [n]$. In summary, each client's secret key is composed of the terms $(\mathsf{imsk}_i, K_i, a_i)$, and the master secret key is simply the union of all clients' secret keys.
- **KGen**: an authority with the master secret key can compute a functional key for the vector $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n) \in \mathbb{Z}_q^{m \cdot n}$ as follows where $\rho \xleftarrow{\$} \mathbb{Z}_q$ is fresh randomness:

$$\{\mathsf{isk}_i := \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_i, \widetilde{\mathbf{y}}_i)\}_{i \in [n]} \text{ where } \widetilde{\mathbf{y}}_i = (\mathbf{y}_i, 0^m, \rho, -\rho a_i, 0)$$

- **Enc**: for client $i \in [n]$ to encrypt $\mathbf{x}_i \in \mathbb{Z}_q^m$ to some label t, it samples $\mu_{i,t} \xleftarrow{\$} \mathbb{Z}_q$ if it has not been sampled before, and outputs the following ciphertext:

$$\mathsf{IPE}.\mathbf{Enc}\,(\mathsf{imsk}_i, \widetilde{\mathbf{x}}_i) \text{ where } \widetilde{\mathbf{x}}_i = (\mathbf{x}_i, 0^m, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i \mu_{i,t}, \mu_{i,t}, 0)$$

- **Dec**: to decrypt, simply use each isk_i to decrypt the ciphertext ct_i from the i-th client and obtain a partial decryption p_i; then, output the discrete log of $\prod_{i \in [n]} p_i$. Since decryption requires computing discrete logarithm, the outcome of the inner-product computation must lie within a polynomially-bounded space for the decryption to be efficient.

We now show correctness. Suppose that $\mathsf{ct}_1, \ldots, \mathsf{ct}_n$ are n honestly generated ciphertexts all for the same label t, and for plaintext vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$, respectively. Further, suppose that $(\mathsf{isk}_1, \ldots, \mathsf{isk}_n)$ is the functional key for the vector $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n)$. Then, applying isk_i to ct_i gives the partial decryption result

$$\begin{aligned} p_i &= [\![\langle \mathbf{x}_i, \mathbf{y}_i \rangle + \rho \cdot \mathsf{CPRF.Eval}(K_i, t) + \rho \cdot a_i \mu_{i,t} - \rho a_i \cdot \mu_{i,t}]\!]_T \\ &= [\![\langle \mathbf{x}_i, \mathbf{y}_i \rangle + \rho \cdot \mathsf{CPRF.Eval}(K_i, t)]\!]_T \end{aligned}$$

Therefore, when we compute the product $\prod_{i \in [n]} p_i$, the part related to the CPRF all cancel out, leaving us the term $[\![\mathbf{x}, \mathbf{y}]\!]_T$ where $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

Intuition. In comparison with the strawman scheme in Section 2.2, here we introduce the additional term $a_i \mu_{i,t}$ to protect the randomizing term $\mathsf{CPRF}.\mathbf{Eval}(K_i$ $, t)$ in the ciphertext, where a_i is part of the master secret key for client i. We also introduce the extra term $[\![\mu_{i,t}]\!]$ to client i's ciphertext component, and the extra term $[\![-\rho a_i]\!]$ to client i's key component, where ρ shared across all clients' key vectors but fresh for each key. These terms make sure that the newly introduced $a_i \mu_{i,t}$ term would cancel out during decryption, such that each client's partial decryption result is preserved as in the strawman scheme.

Table 2: Sequence of hybrids, where $\star$ denotes the most technical step to be elaborate later. Here we show the vectors passed to the underlying IPE's **Enc** and **KGen** functions in each hybrid. Q_{kgen} denotes the maximum number of **KGen** queries made by the adversary. For conciseness, we write $\mathsf{CPRF}(K_i, t)$ as a shorthand for $\mathsf{CPRF}.\mathbf{Eval}(K_i, t)$. Note that the ρ term is sampled fresh at random for each **KGen** query.

Hybrid	**Enc**	**KGen**	assumption
Real_1	$\left(\mathbf{x}_i^{(1)}, \mathbf{0}, \mathsf{CPRF}(K_i,t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right)$	$\left(\mathbf{y}_i^{(1)}, \mathbf{0}, \rho, -\rho a_i, 0\right)$	
Hyb_0	$\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}(K_i,t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right)$	$\left(\mathbf{y}_i^{(1)}, \mathbf{0}, \rho, -\rho a_i, 0\right)$	FH-IND of IPE
Hyb_ℓ $\ell \in [Q_{\text{kgen}}]$	same as Hyb_0	first ℓ: $\left(\mathbf{0}, \mathbf{y}_i^{(0)}, \rho, -\rho a_i, 0\right)$ remaining: $\left(\mathbf{y}_i^{(1)}, \mathbf{0}, \rho, -\rho a_i, 0\right)$	explained below $\star$
Hyb^*	$\left(\mathbf{0}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}(K_i,t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right)$	$\left(\mathbf{0}, \mathbf{y}_i^{(0)}, \rho, -\rho a_i, 0\right)$	FH-IND of IPE
Real_0	$\left(\mathbf{x}_i^{(0)}, \mathbf{0}, \mathsf{CPRF}(K_i,t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right)$	$\left(\mathbf{y}_i^{(0)}, \mathbf{0}, \rho, -\rho a_i, 0\right)$	FH-IND of IPE

In our security proof, we will rely on the security of IPE to hide the $[\![-\rho^* \cdot a_i]\!]_{i\in\mathcal{H}}$ terms pertaining to honest clients $\mathcal{H}$ from some "challenge key" whose shared randomness is ρ^*, and instead move information about $\{[\![\rho^* \cdot \mathsf{CPRF}.\mathbf{Eval}(K_i, t)]\!]\}_{i\in\mathcal{H}, t=1,2,3,\ldots}$ to the honest clients' ciphertext components (see hybrid $\mathsf{H}_{\ell-1,1}$ in Section 2.4). We then argue that under the Decisional Linear assumption, the terms $\{[\![\rho^* \cdot \mathsf{CPRF}.\mathbf{Eval}(K_i, t)]\!]\}_{i\in\mathcal{H}, t=1,2,3,\ldots}$ are computationally indistinguishable from random terms such that for each t their product is conserved (see hybrid $\mathsf{H}_{\ell-1,3}$ of Section 2.4). Moreover, the above should hold even when the adversary may have information about $[\![\rho^*]\!]$ (from knowledge of the challenge key and corrupt clients' master secret keys), $\{[\![\mathsf{CPRF}.\mathbf{Eval}(K_i, t)]\!]\}_{i\in\mathcal{H}, t=1,2,3,\ldots}$ (from honest clients' ciphertexts), and $\{[\![\rho \cdot a_i]\!]\}_{i\in\mathcal{H}}$ for any ρ contained in a non-challenge key (from knowledge of non-challenge keys).

2.4 Proving Selective Function-Hiding Security

We first describe how to prove *selective*, function hiding security, assuming that the underlying IPE scheme satisfies *selective*, function-hiding, indistinguishability-based security, the CPRF scheme is secure, and that the Decisional Linear problem is computationally hard. Later in Section 2.5, we discuss the additional techniques needed for proving adaptive security.

To prove that our scheme satisfies selective function-hiding indistinguishability-based security, we need to go through a sequence of hybrids as shown in Table 2. Note that Table 2 shows only how the challenger generates ciphertext and key components for an *honest* client $i \in [n]$. For a *corrupted* client i, the security game stipulates that $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{(1)}$ and $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$, and thus the challenger simply runs the honest **Enc** or **KGen** algorithm as in the real world.

The steps where we apply the function-hiding indistinguishability security (denoted FH-IND in Table 2) of the underlying IPE are relatively straightforward. The most technical part in the proof is to argue that $\mathsf{Hyb}_{\ell-1}$ is computationally indistinguishable from Hyb_ℓ for $\ell \in [Q_{\text{kgen}}]$, where we are switching the keys queries one by one from world 1 to world 0. In $\mathsf{Hyb}_{\ell-1}$, the first $\ell - 1$ key queries are answered using $\mathbf{y}^{*(0)}$, whereas the remaining are answered using $\mathbf{y}^{*(1)}$. We want to switch the ℓ-th key query to using $\mathbf{y}^{*(0)}$ instead which will lead to Hyb_ℓ. To this end, we carry out another sequence of inner hybrids as shown in Table 3. We first rely on the security of the IPE scheme to accomplish the following (see the experiment $\mathsf{H}_{\ell-1,1}$):

1. move information about $[\![\mathsf{CPRF}(K_i, t)\rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$ from the key to the honest ciphertexts, where ρ^* denotes the shared randomness in the challenge key query; and
2. remove information about $[\![\rho^* a_i]\!]_{i \in \mathcal{H}}$ from the challenge key.

At this moment, we can switch the $[\![\mathsf{CPRF}(K_i, t)\rho^*]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$ terms in the honest ciphertexts to random denoted $[\![T_{i,t}]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$ (subject to the constraint that their product is preserved in each time step t), and uncorrelate these terms with the other ciphertext terms containing information about $\mathsf{CPRF}(K_i, t)$. This can be accomplished through a reduction to the security of the CPRF and the Decisional Linear assumption (hybrids $\mathsf{H}_{\ell-1,2}$ and $\mathsf{H}_{\ell-1,3}$). The Decisional Linear step is arguably the most technical step in our selective security proof, and we provide the detailed proof in Claim 4 in the subsequent formal sections (see also the intuition in Section 2.3).

At this moment, we can switch the terms $[\![T_{i,t} + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$ contained in the honest ciphertexts to $[\![T_{i,t} + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)} \rangle]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$ through an information theoretic step. For this step to hold, we rely on the admissibility rule imposed on the adversary, that is, for any honest plaintexts $\left\{ \left(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)} \right) \right\}_{i \in \mathcal{H}}$ queried for the same label t, and for any pair of key vectors queried $\left(\mathbf{y}^{(0)}, \mathbf{y}^{(1)} \right)$,

$$\left\langle \{\mathbf{x}_i^{(0)}\}_{i \in \mathcal{H}}, \{\mathbf{y}_i^{(0)}\}_{i \in \mathcal{H}} \right\rangle = \left\langle \{\mathbf{x}_i^{(1)}\}_{i \in \mathcal{H}}, \{\mathbf{y}_i^{(1)}\}_{i \in \mathcal{H}} \right\rangle \tag{4}$$

This admissibility rule implies that if for some $i \in \mathcal{H}$, the pair $(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(1)})$ and the pair $(\widetilde{\mathbf{x}}_i^{(0)}, \widetilde{\mathbf{x}}_i^{(1)})$ were queried on the same label t, then the following must hold for any key pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ queried:

$$\left\langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)} \right\rangle - \left\langle \widetilde{\mathbf{x}}_i^{(0)}, \mathbf{y}_i^{(0)} \right\rangle = \left\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)} \right\rangle - \left\langle \widetilde{\mathbf{x}}_i^{(1)}, \mathbf{y}_i^{(1)} \right\rangle \tag{5}$$

Finally, we can go through symmetric steps mirroring the first half of proof, and eventually arrive at Hyb_ℓ.

Table 3: Selective security: inner hybrids to go from $\mathsf{Hyb}_{\ell-1}$ to Hyb_{ℓ}. $\mathbf{y}^{*(b)} := (\mathbf{y}_1^{*(b)}, \ldots, \mathbf{y}_n^{*(b)})$ for $b \in \{0,1\}$ denote the key vectors submitted in the ℓ-th **KGen** query, and ρ^* is the randomness used in the ℓ-th **KGen** query.

Hybrid		assumption
$\mathsf{Hyb}_{\ell-1}$	see Table 2	
$\mathsf{H}_{\ell-1,1}$	**Enc** : $\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}(K_i,t) + a_i\mu_{i,t}, \mu_{i,t}, \mathsf{CPRF}(K_i,t)\cdot\rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)}\rangle\right)$ **KGen** : first $\ell-1$: $\left(0^m, \mathbf{y}_i^{(0)}, \rho, -\rho a_i, 0\right)$ ℓ-th: $(0^m, 0^m, 0, 0, 1)$ remaining: $\left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i, 0\right)$	FH-IND of IPE
$\mathsf{H}_{\ell-1,2}$	**Enc** : $\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, R_{i,t}\cdot\rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)}\rangle\right)$ where $\sum_{i\in\mathcal{H}} R_{i,t} = -\sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i,t)$ **KGen** : same as $\mathsf{H}_{\ell-1,1}$	correlated pseudorand. of CPRF
$\mathsf{H}_{\ell-1,3}$	**Enc** : $\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, T_{i,t} + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)}\rangle\right)$ where $\sum_{i\in\mathcal{H}} T_{i,t} = -\rho^*\cdot\sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i,t)$ **KGen** : same as $\mathsf{H}_{\ell-1,1}$	DLin
$\mathsf{H}'_{\ell-1,3}$	**Enc** : $\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, T_{i,t} + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)}\rangle\right)$ where $\sum_{i\in\mathcal{H}} T_{i,t} = -\rho^*\cdot\sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i,t)$ **KGen** : same as $\mathsf{H}_{\ell-1,1}$	identically distributed
$\mathsf{H}'_{\ell-1,2}$	**Enc** : $\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, R_{i,t}\cdot\rho^* + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)}\rangle\right)$ where $\sum_{i\in\mathcal{H}} R_{i,t} = -\sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i,t)$ **KGen** : same as $\mathsf{H}_{\ell-1,1}$	DLin
$\mathsf{H}'_{\ell-1,1}$	**Enc** : $\left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}(K_i,t) + a_i\mu_{i,t}, \mu_{i,t}, \mathsf{CPRF}(K_i,t)\cdot\rho^* + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)}\rangle\right)$ **KGen** : same as $\mathsf{H}_{\ell-1,1}$	correlated pseudorand. of CPRF
Hyb_{ℓ}	see Table 2	FH-IND of IPE

2.5 Achieving Adaptive Function-Hiding Security

We need additional techniques for proving adaptive security. To aid understanding, it helps to first observe why our previous proof is inherently selective. In a critical step (i.e., from $\mathsf{Hyb}_{\ell-1}$ to Hyb_{ℓ}) where we switched the challenge key (i.e., the ℓ-th key query) from $(\mathbf{y}_i^{*(1)}, \mathbf{0}, \rho^*, -\rho^* a_i, 0)$ to $(\mathbf{0}, \mathbf{y}_i^{*(0)}, \rho^*, -\rho^* a_i, 0)$, we need to go through an inner hybrid experiment where we remove information about $\{\rho^* a_i\}_{i\in\mathcal{H}}$ from the honest clients' key components, and instead encode information about $\{\mathsf{CPRF}(K_i, t)\cdot\rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)}\rangle\}_{i\in\mathcal{H}}$ into the ciphertexts (see the experiment $\mathsf{H}_{\ell-1,1}$). In this step, we made use of the fact that the challenger knows the challenge key vector $\mathbf{y}^*$ upfront.

In adaptive security, the adversary need not commit to all the key queries upfront. A naïve approach to prove adaptive security is via complexity leveraging, i.e., the challenger guesses the challenge key query upfront, and abort the experiment if the guess turns out to be wrong later. The problem with this approach is that it incurs exponential loss in the security failure, and therefore we would have to make the underlying computationally assumptions secure against sub-exponential time adversaries to absorb this security loss. By contrast, our approach does not incur such a loss in security, and we can thus reduce the adaptive function-hiding security of our MCIPE scheme to standard assumptions against polynomial-time adversaries.

Specifically, we show that the scheme described in Section 2.3, when instantiated with a *particular* IPE scheme that satisfies adaptive, function-hiding indistinguishable security, the resulting MCIPE scheme would indeed satisfy function-hiding, adaptive security. To prove this, we can no longer treat the underlying IPE as a blackbox as in our selective security proof. We need to completely unwrap the construction and rely on properties of the specific IPE employed to prove adaptive security. Our proof techniques are inspired by those of Abdalla et al. [5], who constructed an adaptively secure, *multi-input* inner-product encryption (MIIPE). MIIPE can be considered as a special case of MCIPE where *all ciphertexts have the same label* (or time step). This relaxation makes it easier to construct MIIPE. Therefore, the adaptive function-hiding MI-IPE scheme by Abdalla et al. [5] does not easily imply a multi-client counterpart. In particular, for MCIPE, unless we are willing to tolerate linear in n ciphertext size per client, all known *non-lattice-based* constructions require RO, *even for function-revealing* constructions [4, 10, 15].

Our adaptively secure scheme. Concretely, we first apply the function-privacy upgrade of Lin [23] to an adaptively secure, function-revealing IPE scheme of Abdalla et al. [5], resulting in an adaptively secure, weak-function-hiding IPE scheme. We then use the resulting IPE scheme to instantiate our MCIPE scheme described in Section 2.3. The resulting MCIPE scheme, when unwrapped, is as follows — it turns out that we will not need the last slot in the ciphertexts and keys for each client in our adaptive proof, so we remove it from this construction:

– **Setup**: we generate $a_i \xleftarrow{\$} \mathbb{Z}_q$ and random matrices $\mathbf{A}_i, \mathbf{B}_i \xleftarrow{\$} \mathbb{Z}_q^{(k+1)\times k}$ of full rank k, $\mathbf{U}_i \xleftarrow{\$} \mathbb{Z}_q^{(2m+2)\times(k+1)}$, $\mathbf{V}_i \xleftarrow{\$} \mathbb{Z}_q^{(2m+k+3)\times(k+1)}$ for each client $i \in [n]$. We also run the setup algorithm of the CPRF, and obtain $K_1, \ldots, K_n$. In summary, each client's secret key is composed of the terms $(\mathbf{A}_i, \mathbf{B}_i, \mathbf{U}_i, \mathbf{V}_i, K_i, a_i)$, and the master secret key is simply the union of all clients' secret keys.

- **KGen**: an authority with the master secret key can compute a functional key for the vector $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_n) \in \mathbb{Z}_q^{m \cdot n}$ as follows where $\widetilde{\mathbf{y}}_i = (\mathbf{y}_i, 0^m, \rho, -\rho a_i)$ for some fresh random $\rho \xleftarrow{\$} \mathbb{Z}_q, \mathbf{t}_i \xleftarrow{\$} \mathbb{Z}_q^k$:

$$\left\{ [\![\mathbf{d}_i]\!] = [\![\left(\mathbf{I}, \mathbf{U}_i\right)^T \widetilde{\mathbf{y}}_i + \mathbf{V}_i \mathbf{B}_i \mathbf{t}_i]\!], [\![\mathbf{d}'_i]\!] = [\![-\mathbf{B}_i \mathbf{t}_i]\!] \right\}_{i \in [n]}$$

- **Enc**: for client $i \in [n]$ to encrypt a vector $\mathbf{x}_i \in \mathbb{Z}_q^m$ to some label t, it samples $\mu_{i,t} \xleftarrow{\$} \mathbb{Z}_q$ if it has not been sampled before, and outputs the following:

$$\left([\![\mathbf{c}_i]\!] = [\![\left((\widetilde{\mathbf{x}}_i + \mathbf{U}_i \mathbf{A}_i \mathbf{s}_i)^T, (-\mathbf{A}_i \mathbf{s}_i)^T\right)^T]\!], [\![\mathbf{c}'_i]\!] = [\![\mathbf{V}_i^T \mathbf{c}_i]\!] \right)$$
$$\text{where } \widetilde{\mathbf{x}}_i = (\mathbf{x}_i, 0^m, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i \mu_{i,t}, \mu_{i,t})$$

- **Dec**: to decrypt, simply compute $e\left([\![\mathbf{c}_i^T]\!], [\![\mathbf{d}_i]\!]\right) \cdot e\left([\![\mathbf{c}_i'^T]\!], [\![\mathbf{d}'_i]\!]\right)$ for the i-th client and obtain a partial decryption p_i; then, output the discrete log of $\prod_{i \in [n]} p_i$. Since decryption requires computing discrete logarithm, the outcome of the inner-product computation must lie within a polynomially-bounded space for the decryption to be efficient.

Proof roadmap for adaptive security. In our adaptive proof, the outer hybrids remain the same as in Table 2 except that we now need the underlying IPE scheme to have adaptive function-hiding security for make the switches. To switch from $\mathsf{Hyb}_{\ell-1}$ to Hyb_ℓ, we can no longer rely on the previous sequence of inner hybrids (Table 3). Instead, we provide a new sequence of inner hybrids outlined in Table 4.

As shown in Table 4, there are a couple important differences between the previous selective proof and our new adaptive proof. In the selective proof, we switch the challenge key query to IPE functional keys of the vector $(0^m, 0^m, 0, 0, 1)$. This allowed us to erase not just information about $\{\rho^* a_i\}_{i \in \mathcal{H}}$, but also information about the challenge vector $\{\mathbf{y}_i^{*(1)}\}_{i \in \mathcal{H}}$ from the challenge key. Instead, this information is moved to the honest ciphertexts reflected in the terms $[\![\mathsf{CPRF}(K_i, t) \cdot \rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$. But this would require the challenger to know the challenge vector in advance, which we now want to avoid.

In our adaptive proof, we instead switch the challenge key query to IPE functional keys of the vector $(\mathbf{y}_i^{*(1)}, 0^m, 0, 0)$. Here we only remove information about $\{\rho^* a_i\}_{i \in \mathcal{H}}$ from the challenge key, but we retain information about the challenge vector $\{\mathbf{y}_i^{*(1)}\}_{i \in \mathcal{H}}$. Therefore, we only move the terms $[\![\mathsf{CPRF}(K_i, t) \cdot \rho^*]\!]_{i \in \mathcal{H}, t=1,2,3,\ldots}$ to the honest ciphertexts, and the challenger need not know the challenge key vector in advance to do so. Not only so, here, to make this switch, we rely on the structure of the underlying IPE in a non-blackbox fashion (see hybrids $\mathsf{H}_{\ell-1,1}$ and $\mathsf{H}_{\ell-1,2}$). At this moment, we switch the challenge key from using $(\mathbf{y}_i^{*(1)}, 0^m, 0, 0)$ to $(0^m, \mathbf{y}_i^{*(0)}, 0, 0)$ for all $i \in \mathcal{H}$. To make this switch, we make non-blackbox usage of the structure of the underlying IPE, and argue that this switch can be made without affecting the distribution at all, i.e., $\mathsf{H}_{\ell-1,4}$ and $\mathsf{H}'_{\ell-1,4}$ are *identically distributed*, as long as the adversary satisfies the admissibility rule stated in Equation (4) which also implies Equation (5). The rest of the proof takes mirroring steps as the first half to eventually reach hybrid Hyb_ℓ.

The formal proof of adaptive, function-hiding security will be presented in Appendix B.4 of the online full version.

Why we use IPE in a non-blackbox way. In our hybrid sequence for both selective and adaptive proofs, at some point of time we need to switch the inner products from $\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle$ to $\langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)} \rangle$. This step cannot rely on the function-hiding security of IPE because it is possible that $\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle \neq \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)} \rangle$ for some honest user $i \in \mathcal{H}$. So, our idea is to make this switch in a way that the two resulting distributions are identically distributed ($\mathsf{H}_{\ell-1,3}$ to $\mathsf{H}'_{\ell-1,3}$ in the selective proof in Table 3 and $\mathsf{H}_{\ell-1,4}$ to $\mathsf{H}'_{\ell-1,4}$ in the adaptive proof in Table 4). To make this switch in the selective proof, we first switch to a hybrid ($\mathsf{H}_{\ell-1,3}$ in Table 3) in which $\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle$ is in the ciphertext, where we rely on the external randomizing terms $T_{i,t}$ to mask $\langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle$. However, using this technique means we have to know the key queries upfront.

In our adaptive proof, we need to find another way for the proof to go through without knowledge of the key queries. The key step is going from $\mathsf{H}_{\ell-1,4}$ to $\mathsf{H}'_{\ell-1,4}$ in Table 4, where we switch the key from $(\mathbf{y}_i^{*(1)}, 0^m, \ldots)$ to $(0^m, \mathbf{y}_i^{*(0)}, \ldots)$. Here, we argue that making the switch does not affect the distribution if the admissibility rule holds — to do so, we rely on the internal randomness inside the (single-input) IPE scheme, since we no longer can leverage the external random masks $T_{i,t}$ as before.

Why Tomida's techniques do not work. We compare with Tomida [28] and explain why their techniques do not work in the online full version.

2.6 Removing the "All-or-Nothing" Admissibility Rule

So far, our scheme is proven secure in an "all-or-nothing" query setting, that is, for every label t, the adversary must either make at least one ciphertext query on behalf of every honest client, or make none such queries at all. Although it is known that one can remove this restriction on the adversary by wrapping the MCIPE ciphertexts inside a layer of "all-or-nothing encryption" [10, 16, 17], we cannot use the existing techniques as is to get adaptive security and succinct ciphertext at the same time. Recall that in an all-or-nothing encryption (AoNE) scheme [10, 16, 17], if one collects n clients' ciphertexts encrypted to the same label t, then all of them can be decrypted. Otherwise if the collection is not complete for some label t, then no ciphertext encrypted to t can be decrypted and all the plaintexts are kept secret.

Unfortunately, previous AoNE constructions [16, 17] are either not efficient in the sense that the per-client ciphertext size grows linearly with respect to the number of parties [17]; or rely on a random oracle [16, 17]. Moreover, it is

Table 4: Adaptive security: inner hybrids to go from $\mathsf{Hyb}_{\ell-1}$ to Hyb_ℓ. $\mathbf{y}^{*(b)} := (\mathbf{y}_1^{*(b)}, \dots, \mathbf{y}_n^{*(b)})$ for $b \in \{0,1\}$ denote the key vectors submitted in the ℓ-th **KGen** query, and ρ^* is the randomness used in the ℓ-th **KGen** query. Values $\mathbf{u}_i$ in $\mathsf{H}_{\ell-1,1}, \dots, \mathsf{H}'_{\ell-1,1}$ and $\mathbf{b}_i^\perp$ in $\mathsf{H}_{\ell-1,2}, \dots, \mathsf{H}'_{\ell-1,2}$ are sampled once $\forall i \in [n]$ at **Setup**.

Hybrid		assumption
$\mathsf{Hyb}_{\ell-1}$	**Enc** : $\mathbf{c}_i = \left((\widetilde{\mathbf{x}}_i + \mathbf{U}_i\mathbf{A}_i\mathbf{s}_i)^T, (-\mathbf{A}_i\mathbf{s}_i)^T\right)^T$, $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i$ where $\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}\right)$ **KGen** : $\mathbf{d}_i = \left(\mathbf{I}, \mathbf{U}_i\right)^T \widetilde{\mathbf{y}}_i + \mathbf{V}_i\mathbf{B}_i\mathbf{t}_i$, $\mathbf{d}'_i = -\mathbf{B}_i\mathbf{t}_i$, where $\widetilde{\mathbf{y}}_i$ is as follows based on which **KGen** query it is: first $\ell - 1$: $\left(0^m, \mathbf{y}_i^{(0)}, \rho, -\rho a_i\right)$, else: $\left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i\right)$	
$\mathsf{H}_{\ell-1,1}$	**Enc** : same as $\mathsf{Hyb}_{\ell-1}$ **KGen** : $\mathbf{d}_i = \left(\mathbf{I}, \mathbf{U}_i\right)^T \widetilde{\mathbf{y}}_i + \mathbf{V}_i(\mathbf{B}_i\mathbf{t}_i + \mathbf{u}_i)$, $\mathbf{d}'_i = -(\mathbf{B}_i\mathbf{t}_i + \mathbf{u}_i)$, where $\mathbf{u}_i \leftarrow \mathbb{Z}_q^{k+1} \setminus span(\mathbf{B}_i)$ and $\widetilde{\mathbf{y}}_i$ is same as $\mathsf{Hyb}_{\ell-1}$	k-MDDH
$\mathsf{H}_{\ell-1,2}$	**Enc** : $\mathbf{c}_i, \widetilde{\mathbf{x}}_i$: same as $\mathsf{H}_{\ell-1,1}$, $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i - (\mathbf{b}_i^\perp)\rho^*\mathsf{CPRF}(K_i, t)$ where $\mathbf{b}_i^\perp \leftarrow orth(\mathbf{B}_i)$ $s.t.$ $\langle \mathbf{u}_i, \mathbf{b}_i^\perp \rangle = 1$ **KGen** : $\mathbf{d}_i, \mathbf{d}'_i$: same as $\mathsf{H}_{\ell-1,1}$ except $\widetilde{\mathbf{y}}_i$ is as follows based on which **KGen** query it is: ℓ-th: $\left(\mathbf{y}_i^{*(1)}, 0^m, 0, 0\right)$, else: same as $\mathsf{H}_{\ell-1,1}$	identically distributed
$\mathsf{H}_{\ell-1,3}$	**Enc** : $\mathbf{c}_i$: same as $\mathsf{H}_{\ell-1,1}$ except $\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}\right)$ $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i - (\mathbf{b}_i^\perp)\rho^* R_{i,t}$ where $\sum_{i\in\mathcal{H}} R_{i,t} = -\sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i, t)$ **KGen** : same as $\mathsf{H}_{\ell-1,2}$	correlated pseudorand. of CPRF
$\mathsf{H}_{\ell-1,4}$	**Enc** : $\mathbf{c}_i, \widetilde{\mathbf{x}}_i$: same as $\mathsf{H}_{\ell-1,1}$, $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i - (\mathbf{b}_i^\perp)T_{i,t}$ where $\sum_{i\in\mathcal{H}} T_{i,t} = -\rho^* \sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i, t)$ **KGen** : same as $\mathsf{H}_{\ell-1,2}$	DLin
$\mathsf{H}'_{\ell-1,4}$	**Enc** : same as $\mathsf{H}_{\ell-1,4}$ **KGen** : $\mathbf{d}_i, \mathbf{d}'_i$: same as $\mathsf{H}_{\ell-1,4}$ except $\widetilde{\mathbf{y}}_i$ is as follows based on which **KGen** query it is: ℓ-th: $\left(0^m, \mathbf{y}_i^{*(0)}, 0, 0\right)$, else: same as $\mathsf{H}_{\ell-1,1}$	identically distributed
$\mathsf{H}'_{\ell-1,3}$	**Enc** : $\mathbf{c}_i, \widetilde{\mathbf{x}}_i$: same as $\mathsf{H}_{\ell-1,1}$, $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i - (\mathbf{b}_i^\perp)\rho^* R_{i,t}$ where $\sum_{i\in\mathcal{H}} R_{i,t} = -\sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i, t)$ **KGen** : same as $\mathsf{H}'_{\ell-1,4}$	DLin
$\mathsf{H}'_{\ell-1,2}$	**Enc** : $\mathbf{c}_i$: same as $\mathsf{H}_{\ell-1,1}$, $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i - (\mathbf{b}_i^\perp)\rho^*\mathsf{CPRF}(K_i, t)$ $\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}\right)$ **KGen** : same as $\mathsf{H}'_{\ell-1,4}$	correlated pseudorand. of CPRF
$\mathsf{H}'_{\ell-1,1}$	**Enc** : $\mathbf{c}_i, \widetilde{\mathbf{x}}_i$: same as $\mathsf{H}_{\ell-1,1}$, $\mathbf{c}'_i = \mathbf{V}_i^T\mathbf{c}_i$ **KGen** : $\mathbf{d}_i, \mathbf{d}'_i$: same as $\mathsf{H}'_{\ell-1,2}$ except $\widetilde{\mathbf{y}}_i$ is as follows based on which **KGen** query it is: ℓ-th: $\left(0^m, \mathbf{y}_i^{*(0)}, \rho^*, -\rho^* a_i\right)$, else: same as $\mathsf{H}_{\ell-1,1}$	identically distributed
Hyb_ℓ	see Table 2	k-MDDH

also not clear how to extend the existing proof techniques (for removing the "all-or-nothing" query restriction) to the *adaptive function-hiding* setting while retaining succinct ciphertext size [1, 16, 17].

We propose new techniques for performing this upgrade without asymptotically blowing up the ciphertext size, without random oracles, while retaining the adaptive function-hiding security. To make this work, we additionally make the following contributions:

- In Appendix C of the online full version, we construct a new, adaptively secure AoNE scheme that achieves succinct ciphertexts, and reduce its security to Decisional Bilinear Diffie-Hellman assumption.
- Even with an adaptively secure AoNE scheme, it turns out to be difficult to directly prove the security of the upgraded scheme in the adaptive function-hiding setting. We overcome this challenge by introducing a stepping stone: we first prove that the upgraded construction satisfies a relaxed notion called adaptive *weak*-function-hiding security. We then rely on standard techniques [23, 27] to upgrade the resulting adaptive *weak*-function-hiding MCIPE scheme to one that satisfies full adaptive function-hiding security.

We defer the detailed exposition of these new techniques to Appendices C and D of the online full version.

3 Definitions: Multi-Client Inner Product Encryption

Henceforth, we use m to denote the number of coordinates encrypted by each client, and use n to denote the number of clients. In a Multi-Client Inner-Product Functional Encryption (MCIPE) scheme, in every time step, each client $i \in [n]$ encrypts a vector $\mathbf{x}_i \in \mathbb{Z}_q^m$ using its private key ek_i. An authority holding a master secret key msk can generate a functional key $\mathsf{sk}_{\mathbf{y}}$ for a vector $\mathbf{y} \in \mathbb{Z}_q^{mn} = (\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n)$ where each $\mathbf{y}_i \in \mathbb{Z}_q^m$. One can now apply the functional key $\mathsf{sk}_{\mathbf{y}}$ to the collection of all n clients' ciphertexts belonging to the same time step, and an evaluation procedure gives the result $\langle \mathbf{x}, \mathbf{y} \rangle$ where $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n)$.

We use a standard notion of selective indistinguishability for multi-client inner-product encryption [10]. In this standard definition, the time step t is generalized and encoded as an arbitrary label, and only ciphertexts encrypted to the same label can be combined during the decryption process. Mix-and-match among ciphertexts encrypted to different labels should be prevented; however, mix-and-match among the same label is allowed. Formally, an MCIPE scheme consists of the following possibly randomized algorithms:

- $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\lambda)$: the parameter generation algorithm $\mathbf{Gen}$ takes in a security parameter λ and chooses parameters pp — we will assume that pp contains a λ-bit long prime number $q \in \mathbb{N}$ and the description of a suitable cyclic group $\mathbb{G}$ of prime order q.
- $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i \in [n]}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m, n)$: takes in the parameters q, $\mathbb{G}$, m, and n, and outputs a public key mpk, a master secret key msk, and n user

secret keys needed for encryption, denoted $\mathsf{ek}_1, \ldots, \mathsf{ek}_n$, respectively. Without loss of generality, henceforth we may assume that mpk encodes pp so we need not write the parameters pp explicitly below.

– $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$: takes in the public key mpk, the master secret key msk, and a vector $\mathbf{y} \in \mathbb{Z}_q^{mn}$, and outputs a functional secret key $\mathsf{sk}_{\mathbf{y}}$.
– $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t)$: takes in the public key mpk, a user secret key ek_i, a plaintext $\mathbf{x}_i \in \mathbb{Z}_q^m$, and a label $t \in \{0,1\}^*$, outputs a ciphertext $\mathsf{ct}_{i,t}$.
– $v \leftarrow \mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$: takes in the public key mpk, the functional secret key $\mathsf{sk}_{\mathbf{y}}$, and a collection of ciphertexts $\{\mathsf{ct}_{i,t}\}_{i\in[n]}$, outputs a decrypted outcome $v \in \mathbb{Z}_q$.

Correctness. For correctness, we require that for any $\lambda \in \mathbb{N}$, for any $\mathsf{pp} := (q, \ldots)$ in the support of $\mathbf{Gen}(1^\lambda)$, the following holds with probability 1 for any $m, n \in \mathbb{N}$: for any $\mathbf{y} \in \mathbb{Z}_q^{mn}$, and any $\mathbf{x} := (\mathbf{x}_1, \ldots, \mathbf{x}_n) \in \mathbb{Z}_q^n$, and any $t \in \{0,1\}^*$: let $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m, n)$, let $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$, let $\mathsf{ct}_{i,t} \leftarrow \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t)$ for $i \in [n]$, and let $v \leftarrow \mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$, it must be that $v = \langle \mathbf{x}, \mathbf{y} \rangle$.

Function-hiding IND-security for MCIPE**.** Consider the following experiment between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

Experiment $\mathsf{MCIPE\text{-}Expt}^b(1^\lambda)$:

– **Setup.** $\mathcal{A}(1^\lambda)$ outputs a set of corrupted parties $\mathcal{K} \subset [n]$, as well as the parameters m and n to the challenger $\mathcal{C}$. The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\lambda)$, and $(\mathsf{mpk}, \mathsf{msk}, \{\mathsf{ek}_i\}_{i\in[n]}) \leftarrow \mathbf{Setup}(\mathsf{pp}, m, n)$; it gives mpk and $\{\mathsf{ek}_i\}_{i\in\mathcal{K}}$ to $\mathcal{A}$.
– **Query.** The adversary can make the following types of queries:
  - **KGen queries.** Whenever the adversary $\mathcal{A}$ makes a **KGen** query with two vectors $\mathbf{y}^{(0)} \in \mathbb{Z}_q^{mn}$ and $\mathbf{y}^{(1)} \in \mathbb{Z}_q^{mn}$: $\mathcal{C}$ calls $\mathsf{sk}_{\mathbf{y}^{(b)}} := \mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y}^{(b)})$ and returns $\mathsf{sk}_{\mathbf{y}^{(b)}}$ to $\mathcal{A}$;
  - **Enc queries.** Whenever $\mathcal{A}$ makes an **Enc** query with the tuple $(i, t, \mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)})$, the challenger $\mathcal{C}$ calls $\mathsf{ct}_{i,t} := \mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_{i,t}^{(b)}, t)$ and returns $\mathsf{ct}_{i,t}$ to $\mathcal{A}$;

An adversary $\mathcal{A}$ is said to be *admissible* iff the following hold with probability 1 where $\mathcal{H} := [n]\backslash\mathcal{K}$ denotes the set of honest clients:

1. for every label $t \in \{0,1\}^*$, either for every $i \in \mathcal{H}$, $\mathcal{A}$ has made at least one **Enc** query of the form $(i, t, \_, \_)$, or $\mathcal{A}$ made no **Enc** query of the form $(i, t, \_, \_)$ for any $i \in \mathcal{H}$.
2. if $\mathcal{A}$ ever makes an **Enc** query with the tuple $(i, t, \mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)})$ for some corrupt $i \in \mathcal{K}$, it must be that $\mathbf{x}_{i,t}^{(0)} = \mathbf{x}_{i,t}^{(1)}$;
3. for any pair $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ submitted in a **KGen** query where for $b \in \{0,1\}$, $\mathbf{y}^{(b)} := (\mathbf{y}_1^{(b)}, \ldots, \mathbf{y}_n^{(b)}) \in \{0,1\}^{mn}$, it must be that

(a) for $i \in \mathcal{K}$, $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$.

(b) for any collection $\{\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)}\}_{i\in\mathcal{H}}$ pertaining to the same t where each pair $(\mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)})$ for $i \in \mathcal{H}$ has been submitted in an **Enc** query of the form $(i, t, \mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)})$,

$$\left\langle (\mathbf{x}_{i,t}^{(0)})_{i\in\mathcal{H}}, (\mathbf{y}_i^{(0)})_{i\in\mathcal{H}} \right\rangle = \left\langle (\mathbf{x}_{i,t}^{(1)})_{i\in\mathcal{H}}, (\mathbf{y}_i^{(1)})_{i\in\mathcal{H}} \right\rangle \quad (6)$$

Definition 1 (Adaptive, function-hiding IND-security of MCIPE). *We say that an* MCIPE *scheme is adaptive, function-hiding IND-secure iff for any non-uniform probabilistic polynomial-time admissible adversary $\mathcal{A}$, its views in* $\mathsf{MCIPE\text{-}Expt}^0(1^\lambda)$ *and* $\mathsf{MCIPE\text{-}Expt}^1(1^\lambda)$ *are computationally indistinguishable.*

Definition 2 (Selective, function-hiding IND-security of MCIPE). *We say that an* MCIPE *scheme is selective, function-hiding IND-secure iff for any non-uniform probabilistic polynomial-time (*PPT*) admissible adversary $\mathcal{A}$ also satisfying an additional constraint that $\mathcal{A}$ always makes all* **KGen** *queries ahead of any* **Enc** *query, its views in* $\mathsf{MCIPE\text{-}Expt}^0(1^\lambda)$ *and* $\mathsf{MCIPE\text{-}Expt}^1(1^\lambda)$ *are computationally indistinguishable.*

Remark 1 (The all-or-nothing admissibility rule). We also call the first admissibility rule the "all-or-nothing" admissibility rule. Jumping ahead, this rule is necessary later to show that the hybrids $\mathsf{H}_{\ell,3}$ and $\mathsf{H}'_{\ell,3}$ are identically distributed. In Appendices C and D.2 of the online full version, we present new techniques for eventually *removing the all-or-nothing admissibility rule*, thus strengthening the security of the scheme.

4 Preliminaries

We review bilinear groups and relevant hardness assumptions in Appendix A of the online full version.

4.1 Function-Hiding (Single-Input) Inner Product Encryption

We will need a single-input inner-product encryption scheme — henceforth we call this building block Inner Production Encryption (IPE). IPE can be viewed as a special case of multi-client inner product encryption when $n = 1$. However, we will need our underlying IPE to satisfy a few nice properties, including the fact that **Enc** and **KGen** should still work when taking in the group encoding of the plaintext or key vector; moreover, we want that the scheme computes the "inner-product in the exponent". Formally, the special IPE scheme we need consists of the following possibly randomized algorithms:

– $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\lambda)$: takes in a security parameter λ and samples public parameters pp. We will assume that pp contains the description of a bilinear group $(\mathbb{G}, \mathbb{G}_T)$ of prime order q, a random generator $g \in \mathbb{G}$, and the description of the pairing operator $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

- $\mathsf{imsk} \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$: takes in the public parameters pp and the dimension m of the plaintext vector, outputs a secret key imsk.
- $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{imsk}, [\![\mathbf{y}]\!])$: takes in the secret key imsk, and a vector of group elements $[\![\mathbf{y}]\!] \in \mathbb{G}^m$ which represents the group encoding of the vector $\mathbf{y} \in \mathbb{Z}_q^m$, outputs a functional (secret) key $\mathsf{sk}_{\mathbf{y}}$.
- $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{imsk}, [\![\mathbf{x}]\!])$: takes in the secret key imsk, a plaintext vector $[\![\mathbf{x}]\!] \in \mathbb{G}^m$ represented in group encoding, and outputs a ciphertext ct.
- $[\![v]\!]_T \leftarrow \mathbf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct})$: takes in the functional key $\mathsf{sk}_{\mathbf{y}}$ and a ciphertext ct, and outputs a decrypted outcome $[\![v]\!]_T$.

Correctness. Correctness requires that for any $\lambda, m \in \mathbb{N}, \mathbf{x}, \mathbf{y} \in \mathbb{Z}_q^m$, the following holds with probability 1: let $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\lambda)$, $\mathsf{imsk} \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$, $\mathsf{sk}_{\mathbf{y}} \leftarrow \mathbf{KGen}(\mathsf{imsk}, [\![\mathbf{y}]\!])$, $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{imsk}, [\![\mathbf{x}]\!])$, $[\![v]\!]_T \leftarrow \mathbf{Dec}(\mathsf{sk}_{\mathbf{y}}, \mathsf{ct})$, then, it must be that $v := \langle \mathbf{x}, \mathbf{y} \rangle$.

Function-hiding security. Consider the following experiment $\mathsf{IPE\text{-}Expt}^b(1^\lambda)$ between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Experiment $\mathsf{IPE\text{-}Expt}^b(1^\lambda)$:

- **Setup.** The challenger $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathbf{Gen}(1^\lambda)$, and $\mathsf{imsk} \leftarrow \mathbf{Setup}(\mathsf{pp}, m)$, and gives pp to $\mathcal{A}$.
- **Query.** $\mathcal{A}$ makes the following types of queries to $\mathcal{C}$:
  - **KGen** queries: the adversary $\mathcal{A}$ submits $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$; the challenger $\mathcal{C}$ computes $\mathsf{sk}_{\mathbf{y}^{(b)}} \leftarrow \mathbf{KGen}(\mathsf{msk}, [\![\mathbf{y}^{(b)}]\!])$ and returns to $\mathcal{A}$ the resulting $\mathsf{sk}_{\mathbf{y}^{(b)}}$.
  - **Enc** queries: the adversary $\mathcal{A}$ submits $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$; the challenger $\mathcal{C}$ computes $\mathsf{ct} \leftarrow \mathbf{Enc}(\mathsf{mpk}, [\![\mathbf{x}^{(b)}]\!])$, and returns ct to $\mathcal{A}$.

An adversary $\mathcal{A}$ is said to be *admissible* iff the following holds with probability 1: for any $(\mathbf{x}^{(0)}, \mathbf{x}^{(1)})$ tuple submitted in an **Enc** query, for any $(\mathbf{y}^{(0)}, \mathbf{y}^{(1)})$ tuple submitted in a **KGen** query, it must be that $\langle \mathbf{x}^{(0)}, \mathbf{y}^{(0)} \rangle = \langle \mathbf{x}^{(1)}, \mathbf{y}^{(1)} \rangle$.

Definition 3 (Adaptive, Function-hiding IND-security of IPE). *We say that the* IPE *scheme satisfies adaptive, function-hiding IND-security, iff for any non-uniform probabilistic polynomial-time (*PPT*) admissible adversary, its views in* $\mathsf{IPE\text{-}Expt}^0$ *and* $\mathsf{IPE\text{-}Expt}^1$ *are computationally indistinguishable.*

Definition 4 (Selective, Function-hiding IND-security of IPE). *We say that the* IPE *scheme satisfies selective, function-hiding IND-security, iff for any non-uniform* PPT *admissible adversary also satisfying an additional constraint that all* **KGen** *queries must be made before any* **Enc** *query, its views in* $\mathsf{IPE\text{-}Expt}^0$ *and* $\mathsf{IPE\text{-}Expt}^1$ *are computationally indistinguishable.*

Prior works [5,27,31] showed how to construct a function-hiding IPE scheme from the Decisional Linear assumption in bilinear groups. The idea is to first construct an IPE scheme without function privacy from Decisional Linear [5, 27, 31] and then apply a function privacy upgrade [5, 23, 27, 31]. The resulting constructions indeed satisfy the aforementioned nice properties that we need.

4.2 Correlated Pseudorandom Function

A correlated pseudorandom function family consists of the following randomized algorithms:

- $(K_1, \ldots, K_n) \leftarrow \mathbf{Gen}(1^\lambda, n, q)$: takes a security parameter 1^λ and the number of users n, some prime q, and outputs the user secret key K_i for each $i \in [n]$.
- $v \leftarrow \mathbf{Eval}(K_i, x)$: given a user secret key K_i and an input $x \in \{0,1\}^\lambda$, output an evaluation result $v \in \mathbb{Z}_q$.

Correctness. For correctness, we require that for any $\lambda \in \mathbb{N}$, any $(K_1, \ldots, K_n)$ in the support of $\mathbf{Gen}(1^\lambda)$, any input $x \in \{0,1\}^\lambda$, the following holds:

$$\sum_{i \in [n]} \mathsf{CPRF}.\mathbf{Eval}(K_i, x) = 0 \mod q$$

Correlated pseudorandomness. We require that for any non-uniform PPT adversary $\mathcal{A}$ who is allowed corrupt $f \leq n-2$ users and obtain their user secret keys, for any subset U of at most $n-f-1$ honest users, for any input x, the evaluations $\{\mathsf{CPRF}.\mathbf{Eval}(K_i, x)\}_{i \in U}$ are computationally indistinguishable from random values, as long as the adversary has not made a query on the input x.

More formally, correlated pseudorandomness is defined as below. Consider a game denoted $\mathsf{CPRF\text{-}Expt}^b\ (1^\lambda, n, q)$ between $\mathcal{A}$ and a challenger $\mathcal{C}$, parameterized by a bit $b \in \{0,1\}$.

- **Setup.** $\mathcal{A}$ submits a set of corrupt nodes $\mathcal{K} \subset [n]$ of size at most $n-2$. Henceforth, let $\mathcal{H} := [n] \backslash \mathcal{K}$. Now, $\mathcal{C}$ runs the honest $(K_1, \ldots, K_n) := \mathsf{CPRF}.\mathbf{Gen}(1^\lambda, n, q)$ algorithm, and gives $\{K_i\}_{i \in \mathcal{K}}$ to $\mathcal{A}$.
- **Queries.** $\mathcal{A}$ can adaptively make queries: for each query, $\mathcal{A}$ submits an input x. If $b = 0$, the challenger $\mathcal{C}$ chooses random $\{v_i\}_{i \in \mathcal{H}} \xleftarrow{\$} \mathbb{Z}_q^{|\mathcal{H}|}$ subject to the condition that $\sum_{i \in \mathcal{H}} v_i + \sum_{j \in \mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}\ (K_j, x) = 0$, and returns $\{v_i\}_{i \in \mathcal{H}}$ to $\mathcal{A}$. Else if $b = 1$, the challenger gives $\{\mathsf{CPRF}.\mathbf{Eval}(K_i, x)\}_{i \in \mathcal{H}}$ to $\mathcal{A}$.

We say that CPRF satisfies correlated pseudorandomness, iff for any n and q, any non-uniform PPT adversary $\mathcal{A}$'s views in $\mathsf{CPRF\text{-}Expt}^0(1^\lambda, n, q)$ and $\mathsf{CPRF\text{-}Expt}^1$ $(1^\lambda, n, q)$ are computationally indistinguishable.

Construction. Several prior works [1, 14] showed how to construct a correlated pseudorandom function from a standard pseudorandom function (PRF). Without loss of generality, we may assume that PRF's output range is $[0, q-1]$. During the setup phase denoted by **Gen**, sample random PRF keys $k_{i,j}$ for all $i < j$, and let $k_{j,i} = k_{i,j}$. Party i's secret key K_i is defined to be the set $\{k_{i,j}\}_{j \in [n], j \neq i}$. The evaluation function $\mathbf{Eval}(K_i, x)$ is defined as follows:

$$\mathbf{Eval}(K_i, x) = \sum_{j \in [n], j \neq i} (-1)^{j<i} \cdot \mathsf{PRF}(k_{i,j}, x) \mod q$$

Prior works [1, 14, 27] proved that this CPRF satisfies correctness and correlated pseudorandomness, assuming the underlying PRF is secure.

5 Function-Hiding MCIPE

In this section, we give our detailed constructions of function-hiding multi-client inner-product encryption schemes and their formal proofs. In Section 5.1 we present the selective function-hiding secure variant and in Appendix B of the online full version we present the adaptive function-hiding secure variant.

5.1 Selective Function-Hiding MCIPE

Detailed Construction Let $\mathsf{IPE} := (\mathbf{Gen}, \mathbf{Setup}, \mathbf{KGen}, \mathbf{Enc}, \mathbf{Dec})$ denote a function-hiding inner-product encryption scheme, and let $\mathsf{CPRF} := (\mathbf{Gen}, \mathbf{Eval})$ denote a correlated pseudorandom function.

Selective Function-hiding, multi-client inner-product encryption

- $\mathbf{Gen}(1^\lambda)$: let $\mathsf{pp} \leftarrow \mathsf{IPE}.\mathbf{Gen}(1^\lambda)$, and output pp.
- $\mathbf{Setup}(\mathsf{pp}, m, n)$:
  - let $(K_1, \ldots, K_n) := \mathsf{CPRF}.\mathbf{Gen}(1^\lambda, n, q)$;
  - for $i \in [n]$: let $\mathsf{imsk}_i \leftarrow \mathsf{IPE}.\mathbf{Setup}(\mathsf{pp}, 2m+3)$, and $a_i \overset{\$}{\leftarrow} \mathbb{Z}_q$;
  - output $\mathsf{mpk} := \mathsf{pp}$, $\mathsf{msk} := \{\mathsf{imsk}_i, a_i\}_{i\in[n]}$, and $\{\mathsf{ek}_i := (\mathsf{imsk}_i, K_i, a_i)\}_{i\in[n]}$.
- $\mathbf{KGen}(\mathsf{mpk}, \mathsf{msk}, \mathbf{y})$:
  - sample $\rho \overset{\$}{\leftarrow} \mathbb{Z}_q$;
  - let $\widetilde{\mathbf{y}}_i = (\mathbf{y}_i, 0^m, \rho, -\rho a_i, 0)$;
  - let $\mathsf{isk}_i \leftarrow \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_i, [\![\widetilde{\mathbf{y}}_i]\!])$, and output $\mathsf{sk}_{\mathbf{y}} := \{\mathsf{isk}_i\}_{i\in[n]}$.
- $\mathbf{Enc}(\mathsf{mpk}, \mathsf{ek}_i, \mathbf{x}_i, t)$:
  - sample $\mu_{i,t} \overset{\$}{\leftarrow} \mathbb{Z}_q$ if $\mu_{i,t}$ has not been sampled before;
  - let $\widetilde{\mathbf{x}}_i = (\mathbf{x}_i, 0^m, \mathsf{CPRF}.\mathsf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, 0)$;
  - call $\mathsf{ct} \leftarrow \mathsf{IPE}.\mathbf{Enc}(\mathsf{imsk}_i, [\![\widetilde{\mathbf{x}}_i]\!])$, and output ct.
- $\mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\mathbf{y}}, \{\mathsf{ct}_{i,t}\}_{i\in[n]})$: let $[\![v]\!]_T := \prod_{i\in[n]} \mathsf{IPE}.\mathbf{Dec}(\mathsf{isk}_i, \mathsf{ct}_i)$, and output $v := \log([\![v]\!]_T)$.

Asymptotic efficiency. We can instantiate the function-hiding IPE using the scheme described in earlier works [5, 27, 31], based on the Decisional Linear assumption. For the underlying IPE scheme, the ciphertext contains $O(m)$ group elements where m is the length of the vector being encrypted. Similarly, each functional key has only $O(m)$ group elements too. The public parameters contain only the group description.

In our MCIPE construction, to encrypt a length-m vector, each client's ciphertext has only $O(m)$ group elements. A functional key for a length $(n \cdot m)$-vector has size $O(n \cdot m)$ group elements. Each client's secret key has size $O(n)$ where the big-O hides terms related to the security parameter. The public parameters contain only the group description.

Theorem 2. *Suppose that the Decisional Linear assumption holds in $\mathbb{G}$,* IPE *satisfies selective, function-hiding IND-security (see Definition 4), and moreover,* CPRF *satisfies correlated pseudorandomness. Then, the above* MCIPE *scheme satisfies selective function-hiding IND-security.*

We next present the proof of Theorem 2.

Proof of Theorem 2 We consider a sequence of outer hybrid experiments summarized as follows:

$$\mathsf{MCIPE\text{-}Expt}^1 \approx_c \mathsf{Hyb}_0 \approx_c \ldots \approx_c \mathsf{Hyb}_\ell \approx_c \ldots \approx_c \mathsf{Hyb}_{Q_{\mathrm{kgen}}} \approx_c \mathsf{Hyb}^* \approx_c \mathsf{MCIPE\text{-}Expt}^0$$

Further, in Lemma 1 to prove $\mathsf{Hyb}_{\ell-1} \approx_c \mathsf{Hyb}_\ell$, we consider a sequence of inner hybrid experiments summarized as follows:

$$\mathsf{Hyb}_{\ell-1} \approx_c \mathsf{H}_{\ell-1,1} \approx_c \mathsf{H}_{\ell-1,2} \approx_c \mathsf{H}_{\ell-1,3} \approx_c \mathsf{H}'_{\ell-1,3} \approx_c \mathsf{H}'_{\ell-1,2} \approx_c \mathsf{H}'_{\ell-1,1} \approx_c \mathsf{Hyb}_\ell$$

Looking ahead, the proof falls short of showing adaptive security and only shows selective security because in the inner hybrids, the challenger embeds the challenge key $y^{*(b)}$ for $b \in \{0,1\}$ inside the ciphertexts and doing this requires the adversary to make all **KGen** queries before any **Enc** query is made.

Experiment $\mathsf{MCIPE\text{-}Expt}^1$**.** This is the real-world experiment, parameterized by $b = 1$. In this experiment, the challenger $\mathcal{C}$ answers **Enc** and **KGen** queries using the following vectors where ρ is freshly chosen for every **KGen** query:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, 0^m, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right), \quad \widetilde{\mathbf{y}}_i = \left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i, 0\right)$$

Experiment Hyb_0**.** Same as $\mathsf{MCIPE\text{-}Expt}^1$ except that for any honest $i \in \mathcal{H}$, the challenger $\mathcal{C}$ answers **Enc** queries using

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right)$$

Claim 1 *If the* IPE *scheme is function-hiding IND-secure, then,* $\mathsf{MCIPE\text{-}Expt}^1$ *and* Hyb_0 *are computationally indistinguishable.*

Proof. Since this modification preserves the inner products $\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i \rangle$ for any pair of encryption and key vectors queried, and for any $i \in \mathcal{H}$, Hyb_0 is indistinguishable from $\mathsf{MCIPE\text{-}Expt}^1$ due to the function-hiding IND-security of the IPE scheme.

Experiment Hyb_ℓ**.** We next define a sequence of hybrid experiments Hyb_ℓ where $\ell \in [Q_{\mathrm{kgen}}]$ where Q_{kgen} denotes an upper bound the number of **KGen** queries made by $\mathcal{A}$. In Hyb_ℓ, for the first ℓ **KGen** queries, the challenger $\mathcal{C}$ uses $\widetilde{\mathbf{y}}_i = \left(0^m, \mathbf{y}_i^{(0)}, \rho, -\rho a_i, 0\right)$ for any honest $i \in \mathcal{H}$, and uses $\widetilde{\mathbf{y}}_i = \left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i, 0\right)$ for any corrupt $i \in \mathcal{K}$. For the remaining $Q_{\mathrm{kgen}} - \ell$ number of **KGen** queries, $\mathcal{C}$ uses $\widetilde{\mathbf{y}}_i = \left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i, 0\right)$ for all $i \in [n]$.

In Lemma 1, we prove that $\mathsf{Hyb}_{\ell-1} \approx_c \mathsf{Hyb}_{\ell}$ for $\ell \in [Q_{\mathrm{kgen}}]$.

Experiment Hyb^*. The challenger $\mathcal{C}$ answers **Enc** and **KGen** queries using the following vectors for any honest $i \in \mathcal{H}$:

$$\widetilde{\mathbf{x}}_i = \left(0^m, \mathbf{x}_i^{(0)}, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right), \quad \widetilde{\mathbf{y}}_i = \left(0^m, \mathbf{y}_i^{(0)}, \rho, -\rho a_i, 0\right)$$

For corrupt $i \in \mathcal{K}$, the challenger $\mathcal{C}$ still uses:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, 0^m, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right), \quad \widetilde{\mathbf{y}}_i = \left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i, 0\right)$$

Claim 2 *If the* IPE *scheme is function-hiding IND-secure, then,* $\mathsf{Hyb}_{Q_{\mathrm{kgen}}}$ *and* Hyb^* *are computationally indistinguishable.*

Proof. Observe that $\mathsf{Hyb}_{Q_{\mathrm{kgen}}}$ and Hyb^* are almost identical except that the first m coordinates in $\widetilde{\mathbf{x}}_i$ are replaced with 0^m for $i \in \mathcal{H}$. Since this modification preserves the inner products $\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i \rangle$ for any pair of encryption and key vectors queried, and for any $i \in \mathcal{H}$, Hyb^* is computationally indistinguishable from $\mathsf{Hyb}_{Q_{\mathrm{kgen}}}$ due to the function-hiding IND-security of the IPE scheme.

Experiment $\mathsf{MCIPE\text{-}Expt}^0$. This is the real-world experiment, parameterized by $b = 0$. In the experiment $\mathsf{MCIPE\text{-}Expt}^0$, the challenger $\mathcal{C}$ answers **Enc** and **KGen** queries using the following vectors:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(0)}, 0^m, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, 0\right), \quad \widetilde{\mathbf{y}}_i = \left(\mathbf{y}_i^{(0)}, 0^m, \rho, -\rho a_i, 0\right)$$

where ρ is freshly chosen for every **KGen** query.

Claim 3 *If the* IPE *scheme is function-hiding IND-secure, then,* Hyb^* *and* MCIPE $\mathsf{\text{-}Expt}^0$ *are computationally indistinguishable.*

Proof. Observe that Hyb^* is computationally indistinguishable from $\mathsf{MCIPE\text{-}Expt}^0$ since for honest $i \in \mathcal{H}$, the inner-product $\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i \rangle$ is preserved for any pair of encryption and key vectors queried; and for corrupt $i \in \mathcal{K}$, recall that our admissibility stipulates that $\mathbf{x}_i^{(0)} = \mathbf{x}_i^{(1)}$ and $\mathbf{y}_i^{(0)} = \mathbf{y}_i^{(1)}$, and thus it makes no difference whether $\mathbf{x}_i^{(0)}, \mathbf{y}_i^{(0)}$ is used or whether $\mathbf{x}_i^{(1)}, \mathbf{y}_i^{(1)}$ is used by $\mathcal{C}$.

Therefore, to complete the proof of Theorem 2, it suffices to prove the following lemma, i.e., the computational indistinguishability of $\mathsf{Hyb}_{\ell-1}$ and Hyb_{ℓ}.

Lemma 1. *Suppose that the Decisional Linear assumption holds in* $\mathbb{G}$, IPE *satisfies selective function-hiding IND-security, and moreover,* CPRF *satisfies correlated pseudorandomness. Then,* $\mathsf{Hyb}_{\ell-1}$ *is computationally indistinguishable from* Hyb_{ℓ} *for any* $\ell \in [Q_{\mathrm{kgen}}]$.

Proof. We consider a sequence of hybrid experiments.

Experiment $\mathsf{H}_{\ell-1,1}$**.** In $\mathsf{H}_{\ell-1,1}$, for any honest $i \in \mathcal{H}$, the challenger $\mathcal{C}$ uses the following vectors to answer **Enc** and **KGen** queries where $\rho^* \xleftarrow{\$} \mathbb{Z}_q$, and we use $\mathbf{y}^{*(0)}$, $\mathbf{y}^{*(1)}$ to denote the key vectors submitted during the ℓ-th **KGen** query:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) \cdot \rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle \right),$$

$$\widetilde{\mathbf{y}}_i = \begin{cases} \left(0^m, \mathbf{y}_i^{(0)}, \rho, -\rho a_i, 0\right) & \text{first } \ell - 1 \textbf{ KGen} \text{ queries} \\ (0^m, 0^m, 0, 0, 1) & \ell\text{-th } \textbf{KGen} \text{ query} \\ \left(\mathbf{y}_i^{(1)}, 0^m, \rho, -\rho a_i, 0\right) & \text{remaining } Q_{\text{kgen}} - \ell \textbf{ KGen} \text{ queries} \end{cases}$$

Above, ρ is freshly chosen for every **KGen** query, and ρ^* corresponds to the randomness chosen for the challenge **KGen** query, i.e., the ℓ-th **KGen** query.

Observe that $\mathsf{H}_{\ell-1,1}$ is almost identical to $\mathsf{Hyb}_{\ell-1}$ except for the above modifications highlighted in blue. Since these modification preserves the inner products $\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i \rangle$ for any pair of encryption and key vectors queried, and for any $i \in \mathcal{H}$, $\mathsf{H}_{\ell-1,1}$ and $\mathsf{Hyb}_{\ell-1}$ are computationally indistinguishable due to the function-hiding IND-security of the IPE scheme.

Observe that in this hybrid, the challenger needs to know challenge key $\mathbf{y}^{*(1)}$ when answering **Enc** queries and hence $\mathcal{A}$ must make all **KGen** queries ahead of any **Enc** query. This is why our proof technique works only for selective security.

Experiment $\mathsf{H}_{\ell-1,2}$**.** Almost identical to $\mathsf{H}_{\ell-1,1}$, except that for each t label that appears first in an **Enc** query, the challenger $\mathcal{C}$ chooses $\{R_{i,t}\}_{i\in\mathcal{H}}$ at random from $\mathbb{Z}_q$ subject to $\sum_{i\in\mathcal{H}} R_{i,t} = -\sum_{i\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, t)$. For honest $i \in \mathcal{H}$, the challenger $\mathcal{C}$ uses the following vector to answer **Enc** queries:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, R_{i,t} \cdot \rho^* + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle \right)$$

Experiment $\mathsf{H}_{\ell-1,2}$ is computationally indistinguishable from $\mathsf{H}_{\ell-1,1}$ due to the correlated pseudorandomness of CPRF.

Experiment $\mathsf{H}_{\ell-1,3}$**.** Almost identical to $\mathsf{H}_{\ell-1,2}$, except that the challenger $\mathcal{C}$ chooses random $\{T_{i,t}\}_{i\in\mathcal{H}}$ subject to $\sum_{i\in\mathcal{H}} T_{i,t} = -\rho^* \cdot \sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i, t)$, and uses the following vector in any **Enc** query for an honest $i \in \mathcal{H}$:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, \; T_{i,t} + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle \right)$$

Claim 4 *Suppose that the Decisional Linear assumption holds in* $\mathbb{G}$*. Then,* $\mathsf{H}_{\ell-1,3}$ *is computationally indistinguishable from* $\mathsf{H}_{\ell-1,2}$*.*

Proof. We will consider a sequence of hybrid experiments over the set of honest clients. Henceforth let d be the number of honest clients, and let $\mathcal{H} := \{i_1, i_2, \ldots, i_d\} \subseteq [n]$ denote the set of honest clients. We define the hybrid G_j as follows where $j \in [d-1] \cup \{0\}$:

– If i is among the first j honest clients, then $\mathcal{C}$ chooses $\widetilde{T}_{i,t}$ at random;

- If i is not among the first j honest clients and $i \neq i_d$, then, the challenger $\mathcal{C}$ chooses $\widetilde{T}_{i,t} = R_{i,t} \cdot \rho^*$;
- For the last honest client $i = i_d$, the challenger $\mathcal{C}$ chooses $\widetilde{T}_{i,t}$ such that

$$\sum_{i \in \mathcal{H}} \widetilde{T}_{i,t} = -\rho^* \sum_{i \in \mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, t)$$

$\mathcal{C}$ uses the following vector when answering **Enc** queries for any honest $i \in \mathcal{H}$:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t},\ \widetilde{T}_{i,t} + \langle \mathbf{x}_i^{(1)}, \mathbf{y}_i^{*(1)} \rangle\right) \tag{7}$$

Observe that G_0 is the same as $\mathsf{H}_{\ell-1,2}$, and G_{d-1} is the same as $\mathsf{H}_{\ell-1,3}$. Therefore, to prove Claim 4, it suffices to prove that any two adjacent hybrids G_j and G_{j+1} are computationally indistinguishable for $j \in \{0, 1, \ldots, d-2\}$. Below, we prove that if the Decisional Linear assumption holds in $\mathbb{G}$, then indeed G_j and G_{j+1} are computationally indistinguishable for $j \in \{0, 1, \ldots, d-2\}$.

Suppose there is an efficient adversary $\mathcal{A}$ that can distinguish G_j and G_{j+1} with non-negligible probability, we show how to construct an efficient reduction $\mathcal{B}$ that can break the Decisional Linear assumption. Let Q_{enc} denote the maximum number of labels t submitted during **Enc** queries. $\mathcal{B}$ obtains an instance $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$ from a Vector Decisional Linear challenger (see Appendix A.1 of the online full version), where $\mathbf{u}, \mathbf{v}, \mathbf{z} \in \mathbb{Z}_q^{Q_{\text{enc}}}$ and $\beta, \gamma \in \mathbb{Z}_q$. $\mathcal{B}$'s task is to distinguish whether $[\![\mathbf{z}]\!] = [\![\gamma(\mathbf{u} + \mathbf{v})]\!]$ or random. $\mathcal{B}$ will now interact with $\mathcal{A}$ and embed this Decisional Linear instance in its answers.

Let $i^* = i_{j+1} \in \mathcal{H}$ be the index of the $(j+1)$-th honest client.

- **Setup**. $\mathcal{B}$ chooses $\xi \in \mathbb{Z}_q$ at random, and implicitly sets $a_{i^*} = \beta^{-1}$ and $a_{i_d} = \xi \cdot \beta^{-1}$, without actually computing them. $\mathcal{B}$ chooses all other terms in the **Setup** algorithm honestly, and gives mpk and $\{\mathsf{ek}_i\}_{i \in \mathcal{K}}$ to $\mathcal{A}$.
- **KGen queries.**
  1. For the first $\ell - 1$ **KGen** queries:
     - for any honest $i \in \mathcal{H}$, $i \neq i^*$ and $i \neq i_d$, $\mathcal{B}$ knows all the terms necessary to compute isk_i.
     - for $i = i^*$, the reduction $\mathcal{B}$ does not know a_{i^*}, but it can replace the terms $[\![\rho, -\rho a_{i^*}]\!]$ with $[\![\beta\rho', -\rho']\!]$ instead where $\rho' \xleftarrow{\$} \mathbb{Z}_q$. It can compute $[\![\beta\rho]\!]$ because it knows $[\![\beta]\!]$ and ρ'. $\mathcal{B}$ can now continue computing $\mathsf{isk}_{i^*} \leftarrow \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_i, [\![0^m, \mathbf{y}_i^{(0)}, \beta\rho', -\rho', 0]\!])$ normally.
     - for $i = i_d$, $\mathcal{B}$ can compute isk_{i_d} in a similar fashion as above, even if it does not know $a_{i_d} = \xi \cdot \beta^{-1}$.
     - for any corrupt $i \in \mathcal{K}$, $\mathcal{B}$ computes isk_i using the original honest algorithm, since it knows all the necessary terms.
  2. For any **KGen** query after the first ℓ queries, the reduction $\mathcal{B}$ can compute functional key just like for the first $\ell - 1$ queries.
  3. For the ℓ-th **KGen** query, $\mathcal{B}$ wants to embed the γ term from the Decisional Linear challenge into the ρ term for this specific functional key. Recall that $\mathcal{B}$ knows only $[\![\gamma]\!]$ but not γ itself.

- For any corrupt $i \in \mathcal{K}$, observe that $\mathcal{B}$ can compute their respective key component isk_i knowing only $[\![\gamma]\!]$ but not γ itself.
- For any honest $i \in \mathcal{H}$, $\mathcal{B}$ computes $\mathsf{isk}_i \leftarrow \mathsf{IPE}.\mathbf{KGen}(\mathsf{imsk}_i, [\![0^m, 0^m, 0, 0, 1]\!])$.

– **Enc queries.** The adversary $\mathcal{A}$ submits $(i, t, \mathbf{x}_{i,t}^{(0)}, \mathbf{x}_{i,t}^{(1)})$. If $i \in \mathcal{K}$, $\mathcal{B}$ can compute the ciphertext normally since it knows all the necessary terms. Below we focus on the case when $i \in \mathcal{H}$. The first time the label t appears in an **Enc** query for some honest $i \in \mathcal{H}$, the reduction $\mathcal{B}$ picks $\{\widetilde{T}_{i,t}\}_{i\in\mathcal{H}}$ as follows, where u_t, v_t, and z_t denote the t-th component of the vector $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{z}$ from the Decisional Linear challenge[3].

(a) If $i \in \mathcal{H}$, $i \neq i^*$, and $i \neq i_d$: $\mathcal{B}$ chooses $\widetilde{T}_{i,t}$ at random if i is among the first j honest clients, else it implicitly lets $\widetilde{T}_{i,t} := R_{i,t} \cdot \gamma$.

(b) If $i = i^*$: $\mathcal{B}$ implicitly chooses

$$R_{i^*,t} + a_{i^*}\mu_{i^*,t} = u_t, \quad \mu_{i^*,t} = -\beta v_t, \quad \widetilde{T}_{i^*,t} = z_t$$

(c) If $i = i_d$: $\mathcal{B}$ samples $\phi \xleftarrow{\$} \mathbb{Z}_q$, and implicitly chooses

$$\mu_{i_d,t} = -\mu_{i^*,t} \cdot \xi^{-1} + a_{i^*}^{-1} \cdot \phi, \; R_{i_d,t} = -\left(\sum_{i\in\mathcal{H}, i\neq i_d} R_{i,t} + \sum_{i\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, t) \right),$$

$$\widetilde{T}_{i_d,t} = -\left(\sum_{i\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + \sum_{i\in\mathcal{H}, i\neq i^*, i\neq i_d} \widetilde{T}_i + z_t \right)$$

For case (a), computing the ciphertext (see Equation 7) is straightforward. For case (b), it is also easy to see that given $\mathcal{B}$'s knowledge of $[\![u_t]\!]$, $[\![\beta v_t]\!]$, and $[\![z_t]\!]$, one can compute the ciphertext in a straightforward way. For case (c), observe the following. Let

$$\nu = -\left(\sum_{i\in\mathcal{H}, i\neq i_d, i\neq i^*} R_{i,t} + \sum_{i\in\mathcal{K}} \mathsf{CPRF}.\mathbf{Eval}(K_i, t) \right);$$

and thus $R_{i_d,t} = \nu - R_{i^*,t}$.

$$\begin{aligned} [\![R_{i_d,t} + a_{i_d}\mu_{i_d,t}]\!] &= [\![\nu - R_{i^*,t} + \xi a_{i^*} \cdot (-\mu_{i^*,t} \cdot \xi^{-1} + a_{i^*}^{-1} \cdot \phi)]\!] \\ &= [\![\nu - R_{i^*,t} - a_{i^*}\mu_{i^*,t} + \xi \cdot \phi]\!] \\ &= [\![\nu - u_t + \xi \cdot \phi]\!] \end{aligned}$$

Further, $[\![\mu_{i_d,t}]\!] = [\![\beta v_t \cdot \xi^{-1} + \beta \cdot \phi]\!]$. Therefore, both $[\![R_{i_d,t} + a_{i_d}\mu_{i_d,t}]\!]$ and $[\![\mu_{i_d,t}]\!]$ can be computed knowing ν, $[\![u_t]\!]$, ξ, ϕ, $[\![\beta v_t]\!]$, and $[\![\beta]\!]$.

[3] For convenience, we may imagine that the labels t have been renamed to be the integers $\{1, 2, \ldots, Q_{\text{enc}}\}$.

Observe that $R_{i^*,t} \cdot \gamma = (u_t - a_{i^*}\mu_{i^*,t})\gamma = (u_t + \beta^{-1} \cdot \beta v_t)\gamma = (u_t + v_t)\gamma$. Therefore, in the Decisional Linear challenge $([\![1]\!], [\![\beta]\!], [\![\gamma]\!], [\![\mathbf{u}]\!], [\![\beta\mathbf{v}]\!], [\![\mathbf{z}]\!])$ obtained by $\mathcal{B}$, if $\mathbf{z} = \gamma(\mathbf{u}+\mathbf{v})$, then $\mathcal{A}$'s view is identically distributed as in G_j. Else $\mathcal{A}$'s view is identically distributed as in G_{j+1}.

We now continue with the proof of Lemma 1.

Experiment $\mathsf{H}'_{\ell-1,3}$. Almost identical to $\mathsf{H}_{\ell-1,3}$, except that the challenger $\mathcal{C}$ uses the following vector in any **Enc** query for an honest $i \in \mathcal{H}$:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t},\; T_{i,t} + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)} \rangle\right)$$

where the terms $\{T_{i,t}\}_{i\in\mathcal{H}}$ are chosen at random subject to $\sum_{i\in\mathcal{H}} T_{i,t} = -\rho^* \cdot \sum_{i\in\mathcal{K}} \mathsf{CPRF}(K_i, t)$.

As long as $\mathcal{A}$ respects the admissibility rule defined in Section 3, $\mathsf{H}_{\ell-1,3}$ and $\mathsf{H}'_{\ell-1,3}$ are identically distributed.

Experiment $\mathsf{H}'_{\ell-1,2}$. Almost identical to $\mathsf{H}_{\ell-1,2}$, except that the challenger $\mathcal{C}$ chooses uses the following vector to answer **Enc** queries:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, R_{i,t} + a_i\mu_{i,t}, \mu_{i,t}, R_{i,t} \cdot \rho^* + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)} \rangle\right)$$

Experiment $\mathsf{H}'_{\ell-1,1}$. Almost identical to $\mathsf{H}_{\ell-1,1}$, except that the challenger $\mathcal{C}$ chooses uses the following vector to answer **Enc** queries:

$$\widetilde{\mathbf{x}}_i = \left(\mathbf{x}_i^{(1)}, \mathbf{x}_i^{(0)}, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) + a_i\mu_{i,t}, \mu_{i,t}, \mathsf{CPRF}.\mathbf{Eval}(K_i, t) \cdot \rho^* + \langle \mathbf{x}_i^{(0)}, \mathbf{y}_i^{*(0)} \rangle\right)$$

Using a symmetric argument as before, we can prove the computational indistinguishability between $\mathsf{H}'_{\ell-1,3}$ and $\mathsf{H}'_{\ell-1,2}$, and between $\mathsf{H}'_{\ell-1,2}$ and $\mathsf{H}'_{\ell-1,1}$.

Finally, $\mathsf{H}'_{\ell-1,1}$ and Hyb_ℓ are computationally indistinguishable due to the function-hiding IND-security of the IPE scheme, since the inner-product $\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{y}}_i \rangle$ is preserved for any pair of encryption and key vectors queried and for $i \in \mathcal{H}$. This concludes the proof of Lemma 1.

References

1. M. Abdalla, F. Benhamouda, and R. Gay. From single-input to multi-client inner-product functional encryption. In *Asiacrypt*, 2019.
2. M. Abdalla, F. Benhamouda, M. Kohlweiss, and H. Waldner. Decentralizing inner-product functional encryption. In *PKC*, volume 11443, pages 128–157, 2019.
3. M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. In *PKC*, 2015.
4. M. Abdalla, F. Bourse, H. Marival, D. Pointcheval, A. Soleimanian, and H. Waldner. Multi-client inner-product functional encryption in the random-oracle model. In *SCN*, 2020.
5. M. Abdalla, D. Catalano, D. Fiore, R. Gay, and B. Ursu. Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. In *CRYPTO*, 2018.

6. M. Abdalla, R. Gay, M. Raykova, and H. Wee. Multi-input inner-product functional encryption from pairings. In *EUROCRYPT*, 2017.
7. M. Abdalla, D. Pointcheval, and A. Soleimanian. 2-step multi-client quadratic functional encryption from decentralized function-hiding inner-product. *Cryptology ePrint Archive*, 2021.
8. S. Agrawal, M. Clear, O. Frieder, S. Garg, A. O'Neill, and J. Thaler. Ad hoc multi-input functional encryption. In *ITCS*, 2020.
9. S. Agrawal, R. Goyal, and J. Tomida. Multi-input quadratic functional encryption from pairings. In *CRYPTO*, 2021.
10. S. Agrawal, R. Goyal, and J. Tomida. Multi-party functional encryption. In *TCC*, 2021.
11. P. Ananth, A. Jain, and A. Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Cryptology ePrint Archive*, 2015.
12. M. Bellare and T. Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In *Eurocrypt*, 2009.
13. N. Bitansky and V. Vaikuntanathan. Indistinguishability obfuscation from functional encryption. *J. ACM*, 65(6), nov 2018.
14. K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth. Practical secure aggregation for privacy-preserving machine learning. In *CCS*, 2017.
15. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Decentralized multi-client functional encryption for inner product. In *ASIACRYPT*, 2018.
16. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with repetition for inner product. *Cryptol. ePrint*, 2018.
17. J. Chotard, E. Dufour Sans, R. Gay, D. H. Phan, and D. Pointcheval. Dynamic decentralized functional encryption. In *CRYPTO*, 2020.
18. A. Escala, G. Herold, E. Kiltz, C. Rafols, and J. L. Villar. An algebraic framework for diffie-hellman assumptions. In *CRYPTO*, 2013.
19. S. Goldwasser, S. D. Gordon, V. Goyal, A. Jain, J. Katz, F. Liu, A. Sahai, E. Shi, and H. Zhou. Multi-input functional encryption. In *Eurocrypt*, 2014.
20. T. Jager. Verifiable random functions from weaker assumptions. In *TCC*, 2015.
21. F. Kitagawa, R. Nishimaki, and K. Tanaka. Obfustopia built on secret-key functional encryption. In *EUROCRYPT*, 2018.
22. B. Libert and R. Titiu. Multi-client functional encryption for linear functions in the standard model from LWE. In *ASIACRYPT*, 2019.
23. H. Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 prgs. In *CRYPTO*, 2017.
24. B. McMahan and D. Ramage. Federated learning: Collaborative machine learning without centralized training data, 2017.
25. K. Nguyen, D. H. Phan, and D. Pointcheval. Multi-client functional encryption with fine-grained access control. In *ASIACRYPT*, 2023.
26. E. Shi, T.-H. H. Chan, E. Rieffel, R. Chow, and D. Song. Privacy-preserving aggregation of time-series data. In *NDSS*, 2011.
27. E. Shi and K. Wu. Non-interactive anonymous router. In *Eurocrypt*, 2021.
28. J. Tomida. Tightly secure inner product functional encryption: Multi-input and function-hiding constructions. *Theoretical Computer Science*, 833:56–86, 2020.
29. A. Ünal. Impossibility results for lattice-based functional encryption schemes. In *Eurocrypt*, page 169–199, 2020.
30. B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt*, 2005.
31. H. Wee. New techniques for attribute-hiding in prime-order bilinear groups. Manuscript, 2016.