Full-Hiding (Unbounded) Multi-Input Inner Product Functional Encryption from the k-Linear Assumption

Pratish Datta, Tatsuaki Okamoto, and Junichi Tomida

NTT Secure Platform Laboratories
Tokyo, 180-8585 Japan
{datta.pratish,okamoto.tatsuaki,tomida.junichi}@lab.ntt.co.jp

Abstract. This paper presents two *non-generic* and *practically efficient* private key *multi-input functional encryption* (MIFE) schemes for the multi-input version of the *inner product* functionality that are the *first* to achieve simultaneous message and function privacy, namely, the *full-hiding* security for a non-trivial multi-input functionality under *well-studied* cryptographic assumptions. Our MIFE schemes are built in bilinear groups of prime order, and their security is based on the standard *k-Linear* (k-LIN) assumption (along with the existence of semantically secure symmetric key encryption and pseudorandom functions). Our constructions support polynomial number of encryption slots (inputs) without incurring any super-polynomial loss in the security reduction. While the number of encryption slots in our first scheme is apriori bounded, our second scheme can withstand an *arbitrary* number of encryption slots. Prior to our work, there was no known MIFE scheme for a non-trivial functionality, even without function privacy, that can support an unbounded number of encryption slots without relying on any heavy-duty building block or little-understood cryptographic assumption.

Keywords: multi-input functional encryption, inner products, full-hiding security, unbounded arity, bilinear maps

1 Introduction

Functional encryption (FE) [12,36] is a new vision of modern cryptography that aims to overcome the potential limitation of the traditional encryption schemes, namely, the so called "all-or-nothing" control over decryption capabilities, i.e., parties holding the legitimate decryption key can recover the entire message encrypted within a ciphertext, whereas others can learn nothing. Specifically, FE offers additional flexibility by supporting restricted decryption keys which enable decrypters to learn specific functions of encrypted messages, without revealing any additional information. More precisely, an FE scheme for a function family $\mathcal{F}$ involves a setup authority which holds a master secret key and publishes public system parameters. An encrypter uses the public parameters (along with a secret encryption key provided by the setup authority in case of a private key

scheme) to encrypt its message m belonging to some supported message space $\mathcal{M}$, creating a ciphertext CT. A decrypter may obtain a private decryption key SK corresponding to some function $f \in \mathcal{F}$ from the setup authority provided the authority deems that the decrypter is entitled for that key. Such a decryption key SK corresponding to certain decryption function f can be used to decrypt a ciphertext CT encrypting some message m to recover $f(m)$. The basic security requirement for an FE scheme is the privacy of encrypted messages against collusion of decrypters, i.e., an arbitrary number of decrypters cannot jointly retrieve any more information about an encrypted message beyond the union of what they each can learn individually.

Multi-input functional encryption (MIFE), introduced by Goldwasser et al. [23], is a generalization of FE to the setting of multi-input functions. An MIFE scheme has several encryption slots, and messages can be encrypted to different slots independently. A MIFE decryption key for an n-input function f simultaneously decrypts a set of n ciphertexts, each of which is encrypted with respect to one of the n input slots associated with f, to unveil the joint evaluation of f on the n messages encrypted within those n ciphertexts. Just like single-input FE the primary security requirement for an MIFE scheme as well is the privacy of encrypted messages against collusion attacks. However, unlike single-input FE, the formalization of this security notion in case of MIFE is somewhat subtle. In their pioneering work, Goldwasser et al. [23] presented a rigorous framework to formally capture message privacy for MIFE, both in the public key and in the private key regimes.

MIFE is particularly useful in scenarios where informations, which need to be processed together during decryption, become available at different points of time or are supplied by different parties. In fact, MIFE can be employed in a wide range of applications pertaining to computation and mining over encrypted data coming from multiple sources. Examples include executing search queries over encrypted data-bases, processing encrypted streaming data, non-interactive differentially private data releases, multi-client delegation of computations to external servers, and many more. All of these applications are in fact relevant in both the public key and the private key regimes.

In view of its countless practical applications, a series of recent works have attempted to construct MIFE schemes based on various cryptographic tools. These constructions can be broadly classified into two categories. The first line of research has tried to build MIFE schemes for general multi-input functionalities, e.g., arbitrary polynomial-size circuits [23, 6, 13, 24, 28] or Turing machines [7]. Unfortunately however, all such MIFE constructions rely on highly strong cryptographic primitives like indistinguishability obfuscation [8,20], single-input FE for general circuits [20,21], or multilinear maps [19,17], neither of which is currently instantiable using efficient building blocks or under well-studied cryptographic assumptions. Consequently, a second line of research have emerged whose focus is to design concretely efficient MIFE schemes based on standard assumptions for specific multi-input functionalities, e.g., comparison [16,31,15] or multi-input inner product [27, 30, 3]. However, majority of the existing works on MIFE have

concentrated merely on achieving the basic security notion, namely, message confidentiality.

Unfortunately, message confidentiality is not sufficient in several advanced applications of FE, rather privacy also needs to be ensured for the functions for which the decryption keys are issued. This is especially important in situations where the decryption functions themselves contain sensitive informations. Consider the following scenario: Suppose a hospital subscribes to an external cloud server for storing medical records of its patients. In order to ensure confidentiality of the records and, at the same time, remotely perform various computations on the outsourced data from time to time, a promising choice for the hospital is to use an FE scheme to encrypt the records locally prior to uploading to the cloud server. Now, suppose the hospital wishes to retrieve the list of all patients who is receiving treatment for a certain chronic disease from the cloud server. For this, the hospital needs to provide the cloud server a decryption key for the corresponding functionality. However, if the FE scheme used by the hospital possesses no function privacy, then the cloud server would get to know the functionality from the decryption key provided by the hospital. Thus, after performing the assigned computation, if the cloud server notices the name of some celebrity in the obtained list of patients, it would at once understand that the particular celebrity is suffering from such a chronic disease, and it may leak the information to the media possibly for financial gain. This is clearly undesirable from the privacy point of view.

In order to address such scenarios, several recent works have studied the notion of function privacy in the context of FE, both in the single-input setting [38,4,14,25,10,11,9,18,39,27,33,32] and in the multi-input setting [13,6,28,32]. Intuitively, function privacy demands that the decryption keys leak no additional information about the functions embedded within them, beyond what is revealed through decryption. However, it has been observed that the extent to which function privacy can be realized differs dramatically between the public key and the private key regimes. In fact, in the public key setting, where anyone can encrypt messages, only a weak form of function privacy can be realized [10, 11, 25]. More precisely, in order to capture function privacy for FE in the public key setting, the framework must assume that the functions come from a certain high-entropy distribution. On the contrary, function-private FE (both the single-input and the multi-input versions) has been shown to possess great potentials in the private key setting, not only as a stand-alone feature, but also as a very useful building block [5, 6, 29, 33, 32, 28]. Consequently, the research on function-private FE has been focused primarily on the private key setting. However, despite of its immense theoretical and practical significance, so far, there are only a handful of function-private FE schemes available in the literature that can be implemented in practice [9,18,39,27,33,32], and all of them have been designed for single-input functions, precisely, inner products. In case of function-private MIFE, the only known concrete construction is the recent one due to Lin [32]. She has constructed a private key function-private MIFE scheme for computing inner products of arbitrary polynomial degree, where standard inner

product is a degree 2 function. However, her construction employs multilinear maps, and thus is currently uninstantiable in practice.

In this work, our goal is to design practical private key function-private MIFE scheme supporting a polynomial number of encryption slots, incurring only polynomial loss in the security reduction. Goldwasser et al. [23] have already shown that private key MIFE for general functionalities supporting a polynomial number of encryption slots is equivalent to full-fledged indistinguishability obfuscation. Hence, it seems impossible to design such highly expressive MIFE scheme without a sub-exponential security loss [22]. In fact, all existing private key MIFE schemes for general functionalities [23,13,6,28] do suffer from at least a quasi-polynomial security loss to support even a poly-logarithmic number of encryption slots. Hence, we concentrate on a specific multi-input functionality that has a wide range of real-life applications, namely, the natural multi-input generalization of the inner product functionality. This functionality has been first considered by Abdalla et al. [3]. Concretely, a multi-input inner product function $f_{\{\vec{y}_\iota\}_{\iota \in S}}$ is associated with a set S of encryption slot indices and vectors $\vec{y}_\iota \in \mathbb{Z}^m$ for all $\iota \in S$. It takes as input a set of vectors $\{\vec{x}_\iota\}_{\iota \in S}$ with the same index set S, where $\vec{x}_\iota \in \mathbb{Z}^m$ for all $\iota \in S$, and outputs $\sum_{\iota \in S} \vec{x}_\iota \cdot \vec{y}_\iota$, where $\vec{x}_\iota \cdot \vec{y}_\iota$ represents the inner product of the vectors $\vec{x}_\iota$ and $\vec{y}_\iota$ over $\mathbb{Z}$. It is required that the norm of each component inner product $\vec{x}_\iota \cdot \vec{y}_\iota$ is smaller than some upper bound $\mathscr{B}$. Observe that this functionality is different from the high-degree inner product functionality considered by Lin [32]. The multi-input inner product functionality captures various important computations arising in the context of data-mining, e.g., computing weighted mean of informations supplied by different parties. Please refer to [3] for a comprehensive exposure of the practical significance of the multi-input inner product functionality.

Abdalla et al. [3] have presented an MIFE scheme for the multi-input inner product functionality described above in the private key setting, using bilinear groups of prime order. Their construction supports a fixed polynomial number of encryption slots and multi-input inner product functions associated with a fixed index set S of polynomial size, as well as incurs only a polynomial loss in the security reduction. Precisely, the index set S in their construction is of the form $S = [n] = \{1, \ldots, n\}$, where n is the number of encryption slots – a polynomial determined at the time of setup, for the multi-input inner product functions. Their construction achieves adaptive message privacy against arbitrary collusion, as per the framework of Goldwasser et al. [23], in the standard model under the well-studied *k-Linear* (k-LIN) assumption [37]. Prior to the work of Abdalla et al. [3], two independent works, namely, [27, 30] were able to realize a two-input variant of their result, of which [27] achieved it in the generic group model. However, none of these constructions guarantee function privacy. In fact, in their paper [3], Abdalla et al. have posed the construction of a function-private MIFE scheme for the multi-input inner product functionality based on the k-LIN assumption in prime order bilinear groups as an open problem.

Our Contributions

In this paper we solve the above open problem. More specifically, we construct two *concretely efficient* standard-model private key MIFE schemes for the multi-input inner product functionality in prime order bilinear groups that are the *first* to achieve *function privacy* under *well-studied* cryptographic assumptions. In fact, our constructions achieve the unified notion of message and function privacy, namely, the *full-hiding* security, formulated by Brakerski et al. [13] in the context of private key MIFE by combining the corresponding notion in the context of private key single-input FE [4, 14] with the framework for message privacy of MIFE [23], under the k-LIN assumption (along with the existence of semantically secure symmetric key encryption and pseudorandom functions). Both of our constructions support polynomial number of encryption slots and are free from any super-polynomial loss in the security reduction. Our MIFE schemes withstands any polynomial number of decryption key queries and any polynomial number of ciphertext queries for each encryption slot. We employ the elegant technique of dual pairing vector spaces (DPVS) introduced by Okamoto and Takashima [34,35], and are implementable using both symmetric and asymmetric bilinear groups. Just like [3], our first construction supports an apriori fixed number of encryption slots and a fixed slot index set for the multi-input inner product functions. These limitations are removed in our second construction. More precisely, our second construction is capable of supporting an apriori *unbounded* number of encryption slots and multi-input inner product functions with *arbitrary* slot index sets of any polynomial size. In fact, this construction is the *first* MIFE scheme for a non-trivial functionality with an unbounded number of encryption slots, built using efficient cryptographic tools and under well-studied complexity assumptions. The only prior MIFE construction which achieves this feature [7] has been designed using heavy machineries and relies on little-understood cryptographic assumption like public-coin differing input obfuscation [26]. Moreover, the MIFE construction of [7] has been developed in public key setting and possesses no function privacy.

Our MIFE constructions are very efficient. When instantiated under the Symmetric External Diffie-Hellman (SXDH) assumption ($k = 1$ version of the k-LIN assumption) and a symmetric key encryption (SKE) whose secret key size is λ bits, the ciphertexts of our bounded MIFE scheme consist of $2m + 3$ group elements and a λ-bit string, while the decryption keys consist of $n(2m + 3)$ group elements. We would like to mention that these group elements are encrypted by SKE. The master secret key comprises of $n(2m + 3)^2$ elements of the underlying finite field and n λ-bit strings. The encryption incurs one encryption of SKE and $2m+3$ exponentiations, while key generation algorithm incurs one encryption of SKE and $n(2m+3)$ exponentiations. On the other hand, the decryption algorithm involves $(n + 1)$ executions of the decryption algorithm of SKE and $n(2m + 3)$ pairing operations followed by an exhaustive search step over a polynomial-size range of possible values. Here, m and n respectively denote the length of the vectors and the size of the index set associated with the multi-input inner product functionality. Observe that these figures are already in close compliance with the

n-fold extension of the most efficient standard-model full-hiding single-input FE construction for inner products known till date, namely, the scheme by Lin [32] (which is also designed under the SXDH assumption). The exhaustive search step in the decryption algorithm is reminiscent of all currently known bilinear-map-based FE constructions for inner products, both in the single-input and in the multi-input settings. In unbounded scheme, the ciphertext size and decryption key size are the same as bounded scheme, while the master secret key consists of two pseudorandom function (PRF) keys and $(2m+3)^2$ elements of the underlying finite field. The encryption incurs two PRF evaluations and $2m+3$ exponentiations, while the key generation algorithm incurs n executions of the encryption algorithm of SKE, $2n$ PRF evaluations, and $n(2m+3)$ exponentiations. The decryption algorithm, on the other hand, involves n executions of the decryption algorithm of SKE and $n(2m+3)$ pairing operations followed by an exhaustive search step similar to the bounded scheme.

Our Techniques

We now explain the principal ideas underlying our MIFE constructions for the multi-input inner product functionality. In order to simplify the exposition, we ignore many technicalities in this overview.

Our bounded-arity scheme: Since, the multi-input inner product functionality is a multi-input generalization of its single-input version, a natural first step is to explore whether we can obtain a private key full-hiding n-input MIFE scheme for inner products by executing n parallel copies of a private key full-hiding FE scheme for inner products. The most efficient such scheme available in the literature is the one due to Lin [32], which is based on the SXDH assumption. However, the construction is built upon the Decisional-Diffie-Hellman (DDH)-based construction of Abdalla et al. [1] and is not readily amenable to the general k-LIN assumption. Moreover, the construction is built in a two step approach, namely, first constructing an FE scheme for inner products achieving only a weaker form of function privacy, and then bootstrapping to the full-hiding security by using the conversion of Lin and Vaikuntanathan [33]. We want to avoid such an approach, rather our goal is to design a direct construction of full-hiding MIFE for multi-input inner products. So, we start with the full-hiding single-input inner-product FE scheme proposed by Tomida et al. [39]. This construction is direct, and while originally presented under a variant of the Decisional Linear (DLIN) assumption, seems naturally generalizable to the k-LIN assumption. Further, in terms of efficiency, this construction is next to the construction of Lin [32] among the standard-model private key function-private FE constructions available in the literature [9, 18, 39, 32]. Besides, this construction has the flexibility of being implementable in both symmetric and asymmetric bilinear groups.

First, let us briefly review the construction and proof idea of Tomida et al. [39]. We assume familiarity with the DPVS framework for the rest of this section. (The background on DPVS is provided in Section 2.3.) The master secret key MSK in the construction of Tomida et al. [39] consists of a pair of dual

orthogonal bases $(\mathbb{B}, \mathbb{B}^*)$ of a $(2m+5)$-dimensional DPVS, where m is the length of the ciphertext and decryption key vectors. Out of the $2m + 5$ dimensions, $m + 4$ dimensions are utilized in the real construction, while the rest are used in performing various hybrid transitions in the security proof. Note that the use of such hidden dimensions is a powerful feature of the DPVS framework, and it has been proven to be instrumental in deducing various complex security proofs in the literature. The ciphertext CT of [39] encrypting an m-dimensional vector $\vec{x}$ is given by $\text{CT} = (\vec{x}, \vec{0}^m, \vec{0}^2, \varphi_1, \varphi_2, 0)_{\mathbb{B}}$, where $\varphi_1, \varphi_2 \xleftarrow{\mathsf{U}} \mathbb{F}_q$. On the other hand, the decryption key SK corresponding to some m-dimensional vector $\vec{y}$ is of the form $\text{SK} = (\vec{y}, \vec{0}^m, \gamma_1, \gamma_2, \vec{0}^2, 0)_{\mathbb{B}^*}$, where $\gamma_1, \gamma_2 \xleftarrow{\mathsf{U}} \mathbb{F}_q$. Here, $(\vec{v})_{\mathbb{W}}$, for any vector $\vec{v}$ with entries in $\mathbb{F}_q$ and any basis $\mathbb{W}$ of a DPVS, signifies the linear combination of the members of $\mathbb{W}$ using the entries of $\vec{v}$ as coefficients. The decryption algorithm works by computing $e(\text{CT}, \text{SK})$ followed by performing an exhaustive search step over a specified polynomial-size range to determine the output. The correctness readily follows by the dual orthogonality property of $(\mathbb{B}, \mathbb{B}^*)$.

Recall that in the full-hiding security experiment for single-input inner product FE [4,14], first the challenger $\mathcal{B}$ sets up the system and samples a random bit $\beta \xleftarrow{\mathsf{U}} \{0, 1\}$. Next, the adversary $\mathcal{A}$ is allowed to adaptively make any polynomial number of ciphertext and decryption key queries to $\mathcal{B}$. In order to make a ciphertext query, $\mathcal{A}$ submits a pair of message vectors $(\vec{x}_0, \vec{x}_1)$ to $\mathcal{B}$, while to make a decryption key query, $\mathcal{A}$ submits a pair of vectors $(\vec{y}_0, \vec{y}_1)$ to $\mathcal{B}$. Depending on the random bit β, $\mathcal{B}$ returns respectively an encryption of $\vec{x}_\beta$ and a decryption key for vector $\vec{y}_\beta$ to the adversary in response to the respective queries. Finally, the adversary has to correctly guess the random bit β to win the experiment. The restriction on the queries of $\mathcal{A}$ is that for all pairs of vectors $(\vec{x}_0, \vec{x}_1)$ for which a ciphertext query is made and for all pairs of vectors $(\vec{y}_0, \vec{y}_1)$ for which a decryption key query is made, it should hold that $\vec{x}_0 \cdot \vec{y}_0 = \vec{x}_1 \cdot \vec{y}_1$.

In order to prove security of the construction of [39] in the above full-hiding model, the following hybrid transitions are performed: The initial hybrid is the real full-hiding experiment with the challenge bit $\beta = 0$, i.e., where the forms of the ciphertexts and decryption keys returned to $\mathcal{A}$ are respectively $\text{CT}^* = (\vec{x}_0, \vec{0}^m, \vec{0}^2, \varphi_1, \varphi_2, 0)_{\mathbb{B}}$ and $\text{SK}^* = (\vec{y}_0, \vec{0}^m, \gamma_1, \gamma_2, \vec{0}^2, 0)_{\mathbb{B}^*}$, while the final hybrid corresponds to the real full-hiding experiment with $\beta = 1$, i.e., where the forms of the ciphertexts and decryption keys returned to the adversary are of the form $\text{CT}^* = (\vec{x}_1, \vec{0}^m, \vec{0}^2, \varphi_1, \varphi_2, 0)_{\mathbb{B}}$ and $\text{SK}^* = (\vec{y}_1, \vec{0}^m, \gamma_1, \gamma_2, \vec{0}^2, 0)_{\mathbb{B}^*}$ respectively. Towards achieving this change, first, applying a combination of a computational change using the DLIN assumption, in conjunction with a conceptual transformation of the underlying bases, the form of the ciphertexts are altered one by one to $\text{CT}^* = (\vec{x}_0, \vec{x}_1, \vec{0}^2, \varphi_1, \varphi_2, 0)_{\mathbb{B}}$. In the next step, applying another combination of computational and conceptual changes, the form of the queried decryption keys are changed one by one to the form $\text{SK}^* = (\vec{0}^m, \vec{y}_1, \gamma_1, \gamma_2, \vec{0}^2, 0)_{\mathbb{B}^*}$. This is the most subtle transition step, and this is where we have to rely crucially on the restriction of the security model. More precisely, observe that before altering the decryption keys, decrypting the queried ciphertexts using the queried de-

cryption keys result in $\vec{x}_0 \cdot \vec{y}_0$, whereas after the transformation, the decryption results are $\vec{x}_1 \cdot \vec{y}_1$. However, thanks to the restriction of the full-hiding security experiment, we can ensure that the decryption results in the two cases are the same, and thus the change cannot be detected through decryption. After this step, the forms of ciphertexts and decryption keys are further altered respectively to $\mathrm{CT}^* = (\vec{x}_1, \vec{x}_0, \vec{0}^2, \varphi_1, \varphi_2, 0)_{\mathbb{B}}$ and $\mathrm{SK}^* = (\vec{y}_1, \vec{0}^m, \gamma_1, \gamma_2, \vec{0}^2, 0)_{\mathbb{B}^*}$, with the help of another conceptual basis transformation. Once this step is executed, the forms of the queried ciphertexts are changed to $\mathrm{CT}^* = (\vec{x}_1, \vec{0}^m, \vec{0}^2, \varphi_1, \varphi_2, 0)_{\mathbb{B}}$ using a reverse transformation to the one used in the first step. Observe that this last step takes us to the experiment corresponding to $\beta = 1$.

Let us now consider an MIFE scheme for the n-input inner product functionality obtained by an n-fold extension of the above single-input scheme. More precisely, consider an n-input MIFE scheme having the following specifications: The master secret key MSK consists of n independently generated master secret keys for the single-input scheme, i.e., $\mathrm{MSK} = \{\mathrm{MSK}_\iota = (\mathbb{B}_\iota, \mathbb{B}_\iota^*)\}_{\iota \in [n]}$. The ciphertext of some vector $\vec{x}_\iota$ with respect to index $\iota \in [n]$ is simply a single-input FE ciphertext for $\vec{x}_\iota$ with respect to MSK_ι, i.e., the ciphertext has the form $\mathrm{CT}_\iota = (\iota, \boldsymbol{c}_\iota = (\vec{x}_\iota, \vec{0}^m, \vec{0}^2, \varphi_{\iota,1}, \varphi_{\iota,2}, 0)_{\mathbb{B}_\iota})$, where $\varphi_{\iota,1}, \varphi_{\iota,2} \xleftarrow{\mathsf{U}} \mathbb{F}_q$. On the other hand, a decryption key associated with a set of n vectors $\{\vec{y}_\iota\}_{\iota \in [n]}$ is given by a set of n decryption keys $\{\mathrm{SK}_\iota\}_{\iota \in [n]}$, where SK_ι is the single-input FE secret key for $\vec{y}_\iota$ with respect to MSK_ι, i.e., $\mathrm{SK} = \{\boldsymbol{k}_\iota = (\vec{y}_\iota, \vec{0}^m, \gamma_{\iota,1}, \gamma_{\iota,2}, \vec{0}^2, 0)_{\mathbb{B}_\iota^*}\}_{\iota \in [n]}$, where $\gamma_{\iota,1}, \gamma_{\iota,2} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for all $\iota \in [n]$. To decrypt a set of n ciphertexts $\{\mathrm{CT}_\iota\}_{\iota \in [n]}$ using a decryption key SK, one first computes $\prod_{\iota \in [n]} e(\boldsymbol{c}_\iota, \boldsymbol{k}_\iota)$, and then performs an exhaustive search step. It is easy to see that the correctness follows analogously to the single-input case.

However, one can readily observe that the above n-input extension is not secure. In particular, the construction leaks partial information. Precisely, notice that for each $\iota \in [n]$, one can easily recover $\vec{x}_\iota \cdot \vec{y}_\iota$ by computing $e(\boldsymbol{c}_\iota, \boldsymbol{k}_\iota)$, whereas ideally one should only be able to learn $\sum_{\iota \in [n]} \vec{x}_\iota \cdot \vec{y}_\iota$. Abdalla et al. [3] also faced a similar challenge while constructing their MIFE scheme by building on a single input inner product FE scheme. In order to overcome this problem, they introduced additional randomness within ciphertexts and decryption keys. Precisely, in order to generate a ciphertext for vector $\vec{x}_\iota$ with respect to index $\iota \in [n]$, they encrypted the vector $(\vec{x}_\iota, z_\iota)$, where $z_1, \ldots, z_n \xleftarrow{\mathsf{U}} \mathbb{F}_q$ are included within the master secret key. Similarly, while preparing a decryption key for a set of n vectors $\{\vec{y}_\iota\}_{\iota \in [n]}$, they sampled a random value $r \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and generated single-input FE decryption keys for the vectors $(\vec{y}_\iota, r)$ for all $\iota \in [n]$, and additionally create the component $k_T = g_T^{\sum_{\iota \in [n]} z_\iota r}$. We attempt to apply their trick to our setting. More precisely, we modify our MIFE construction as follows: We add one additional dimension in the dual orthogonal bases $(\mathbb{B}_\iota, \mathbb{B}_\iota^*)$ for each $\iota \in [n]$, i.e., they are now $(2m + 6)$-dimensional. A ciphertext encrypting the vector $\vec{x}_\iota$ with respect to index $\iota \in [n]$ is of the form

$\mathrm{CT}_\iota = (\iota, \boldsymbol{c}_\iota = (\vec{x}_\iota, \vec{0}^m, z_\iota, \vec{0}^2, \varphi_{\iota,1}, \varphi_{\iota,2}, 0)_{\mathbb{B}_\iota})$, where $z_1, \ldots, z_n \xleftarrow{\cup} \mathbb{F}_q$ are parts of MSK, and the decryption key corresponding to a set of n vectors $\{\vec{y}_\iota\}_{\iota\in[n]}$ is given by $\mathrm{SK} = (\{\boldsymbol{k}_\iota = (\vec{y}_\iota, \vec{0}^m, r, \gamma_{\iota,1}, \gamma_{\iota,2}, \vec{0}^2, 0)_{\mathbb{B}^*_\iota}\}_{\iota\in[n]}, k_T = g_T^{\sum_{\iota\in[n]} z_\iota r})$. Decryption works by first computing $[\prod_{\iota\in[n]} e(\boldsymbol{c}_\iota, \boldsymbol{k}_\iota)]/k_T = g_T^{\sum_{\iota\in[n]} \vec{x}_\iota\cdot\vec{y}_\iota}$, and then performing an exhaustive search step to recover $\sum_{\iota\in[n]} \vec{x}_\iota \cdot \vec{y}_\iota$.

Let us now consider the security of the modified construction. For simplicity, assume that the adversary queries a single decryption key and a single ciphertext for each of the n encryption slots. The full-hiding security model for private key MIFE [13] is an extension of its single-input counter part, but is significantly more complicated compared to it. Analogous to the single-input case, in this multi-input security model, in order to make a ciphertext query for the ι^{th} slot, the adversary has to submit a pair of vectors $(\vec{x}_{\iota,0}, \vec{x}_{\iota,1})$, whereas for making a decryption key query, the adversary has to submit a pair of sets of n vectors $(\{\vec{y}_{\iota,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,1}\}_{\iota\in[n]})$. However, unlike the single-input setting, now the restriction on the queries is that $\sum_{\iota\in[n]} \vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0} = \sum_{\iota\in[n]} \vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}$. Let us try to argue security of our modified construction by taking a similar path to that taken by Tomida et al. [39]. We start with the case where the challenge bit $\beta = 0$, i.e., when the ciphertexts and decryption key returned to the adversary have the form $\mathrm{CT}^*_\iota = (\iota, \boldsymbol{c}^*_\iota = (\vec{x}_{\iota,0}, \vec{0}^m, z_\iota, \vec{0}^2, \varphi_{\iota,1}, \varphi_{\iota,2}, 0)_{\mathbb{B}_\iota})$, for $\iota \in [n]$, and $\mathrm{SK}^* = (\{\boldsymbol{k}^*_\iota = (\vec{y}_{\iota,0}, \vec{0}^m, r, \gamma_{\iota,1}, \gamma_{\iota,2}, \vec{0}^2, 0)_{\mathbb{B}^*_\iota}\}_{\iota\in[n]}, k_T = g_T^{\sum_{\iota\in[n]} z_\iota r})$. Just like [39], first, using a combination of computational changes using the DLIN assumption, in conjunction with a conceptual transformation to the underlying bases, we can alter the forms of all the ciphertexts to $\mathrm{CT}^*_\iota = (\iota, \boldsymbol{c}^*_\iota = (\vec{x}_{\iota,0}, \vec{x}_{\iota,1}, z_\iota, \vec{0}^2, \varphi_{\iota,1}, \varphi_{\iota,2}, 0)_{\mathbb{B}_\iota})$. After this step is done, we would have to change the form of the queried decryption key SK^* so that the first $2m$ coefficients of each $\boldsymbol{k}^*_\iota$ become $(\vec{0}^m, \vec{y}_{\iota,1})$. In order to achieve this change, we first perform a computational change to $\boldsymbol{k}^*_\iota$, for each $\iota \in [n]$, with the help of the DLIN assumption to $\boldsymbol{k}^*_\iota = (\vec{y}_{\iota,0}, \vec{0}^m, r, \gamma_{\iota,1}, \gamma_{\iota,2}, \vec{0}^2, \omega_\iota)_{\mathbb{B}^*_\iota}$, where $\omega_\iota \xleftarrow{\cup} \mathbb{F}_q$ for all $\iota \in [n]$. Next, we need to perform a conceptual transformation to the underlying bases in each slot so that the first two m blocks of each $\boldsymbol{k}^*_\iota$ gets interchanged. However, this conceptual change would generate the term $\vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0} - \vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}$ in the $(2m+6)^{\text{th}}$ coefficient of each ciphertext CT_ι. In the single-input case, such a term vanishes by the restriction on the ciphertext and decryption key queries. But, unlike the single-input case, now $\vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0}$ is not guaranteed to be equal to $\vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}$ for all $\iota \in [n]$, and hence the term in the $(2m+6)^{\text{th}}$ coefficient does not vanish.

In order to overcome this problem, we modify the above construction by introducing a different randomness in each of the n component of the decryption key rather than using a same shared randomness across all the n components. More precisely, a decryption key corresponding to a set of n vectors $\{\vec{y}_\iota\}_{\iota\in[n]}$ has the form $\mathrm{SK} = (\{\boldsymbol{k}_\iota = (\vec{y}_\iota, \vec{0}^m, r_\iota, \gamma_{\iota,1}, \gamma_{\iota,2}, \vec{0}^2, 0)_{\mathbb{B}^*_\iota}\}_{\iota\in[n]}, k_T = g_T^{\sum_{\iota\in[n]} z_\iota r_\iota})$, where

$r_\iota \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for all $\iota \in [n]$. First, observe that this modification does not affect the correctness. Now, with this modification, we can resolve the above problem as follows: In the above conceptual change step, we transform the underlying bases in such a way that not only the first two m blocks of each $\boldsymbol{k}^*_\iota$ gets interchanged, but also each r_ι gets altered to $\widetilde{r}_\iota$, where $\widetilde{r}_\iota = r_\iota + [\vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0} - \vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}]/z_\iota$. Observe that the $\widetilde{r}_\iota$'s are also distributed uniformly and independently over $\mathbb{F}_q$ since r_ι's are so. Also, this new basis transformation will create the additional term $[\vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1} - \vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0}]$ in the $(2m+6)^{\text{th}}$ coefficient of the queried ciphertext in each slot that would cancel out the term $[\vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0} - \vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}]$. Further, notice that $\sum_{\iota \in [n]} z_\iota \widetilde{r}_\iota = \sum_{\iota \in [n]} z_\iota r_\iota$ by the restriction of the full-hiding security experiment of the multi-input setting, namely, $\sum_{\iota \in [n]} \vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0} = \sum_{\iota \in [n]} \vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}$.

Note that our actual construction and security proof, which is presented under the general k-LIN assumption, is more subtle. In our actual construction, we observe that replacing the z_ι values with the scalar 1 and choosing the r_ι values associated with a decryption key under the restriction that $\sum_{\iota \in [n]} r_\iota = 0$ is sufficient to argue the security proof. As a result of this modification, we are able to remove the k_T component from the decryption keys. Also, in the actual construction, we reduce the dimension of the underlying bases further by making a more careful use of the randomness.

Our unbounded-arity scheme: In our bounded-arity scheme, the setup algorithm makes n random dual orthogonal bases for n-input case, and stores them as a master secret key. The first problem is how to make these bases unboundedly from a master secret key, whose size is independent from n. Considering that our scheme is private-key MIPE, to get an idea of making them from a pseudorandom function is not difficult. That is, we prepare a randomly chosen pseudorandom function key as a master secret key in a setup phase, and in encryption or key generation, we can generate dual orthogonal bases from the pseudorandom function with its input being the slot index when they are needed. Actually, this naive idea works in a conditional full-hiding security model, where for each decryption key, all indices included in the decryption keys are queried in ciphertext query. The crucial point is that, for some decryption key queried by the adversary, if all indices that are included in the decryption key are queried in ciphertext query, then all corresponding vectors must satisfy some restrictions to avoid a trivial attack. Concretely, for each decryption key SK_S for a index set S and vectors $\{\vec{y}_\iota\}_{\iota \in S}$, all vectors $\vec{x}_\iota$ for slot $\iota \in S$ queried in ciphertext query, satisfy the following restriction s.t. $\sum_{\iota \in S} \vec{x}_{\iota,0} \cdot \vec{y}_{\iota,0} = \sum_{\iota \in S} \vec{x}_{\iota,1} \cdot \vec{y}_{\iota,1}$. When we construct our bounded-arity scheme, we first construct a scheme that is secure in the conditional full-hiding security model, and then we convert it into one that has full-hiding security with no conditions by a generic transformation, similarly to Abdalla et. al. [3]. We leverage such a restriction in the proof of the underlying scheme.

In the conversion, we prepare a random bit string k_ι for each index. Next, we encrypt all decryption keys and ciphertexts of the underlying scheme with SKE using $K = \bigoplus_{\iota=1}^{n} k_\iota$ as a secret key. Then, we append the random bit string k_ι to ciphertexts for index ι. By the construction, if there exist some indices that are not queried in ciphertext query, an adversary cannot compute K and all ciphertexts and decryption keys are completely hidden from the adversary. Therefore we can exclude such a situation and focus on the conditional full-hiding security model. However, this generic transformation does not work in the unbounded arity-case, because a set of ciphertexts (or indices) needed for decryption differs by each decryption key. Then we do not know how to convert an unbounded-arity scheme secure under the conditional full-hiding security model into one with full-hiding security.

To solve this problem, we introduce a new construction and new proof techniques. Our solution inherits the spirit of the above technique due to Abdalla et. al. [3], but is not completely generic. The basic scheme is that making use of pseudorandom functions as mentioned earlier. Then we introduce another pseudorandom function, which takes an index of slots ι as an input and outputs a random bit string k_ι, which is assigned for each index. Those bit strings are appended to corresponding ciphertexts like the above generic transformation, but we do not encrypt ciphertexts with SKE, or even cannot because it is impossible to decide which indices are needed for decryption in the unbounded case. Instead we encrypt each decryption key with SKE, using the all bit strings corresponding to the index set of decryption key, as a secret key of SKE in some way. We can see that if there are some indices which are not queried in ciphertext query (we call such indices as absent indices), then the decryption keys which contain absent indices will be completely hidden from the adversary. It is because to obtain the secret keys of SKE, the adversary needs all bit strings k_ι (or ciphertexts) for the corresponding indices.

In this construction, however, we cannot use a generic transformation because ciphertexts are not encrypted with SKE. Instead we consider a series of hybrids in the same manner as bounded-arity case for the security proof. During the hybrids, we encounter the problem that there are some decryption keys that have absent indices, and therefore these decryption keys and ciphertexts might not satisfy the restriction as explained above. To solve the problem, we leverage the power of SKE, and it enables us to go the hybrids ahead. More precisely, for the decryption keys that have absent indices, we use the power of SKE, and for the other decryption keys, we use the power of the basic scheme. But here, if we define the secret key of SKE to encrypt a decryption key for a set S as $\bigoplus_{\iota \in S} k_\iota$, likely to the generic transformation of the bounded case, we realize that we cannot make a reduction algorithm for SKE. This problem is mainly due to the flexibleness of decryption keys, that is, a set, which can be associated with secret keys, is not determined in the scheme. Observe that in the bounded case, the set is determined as $\{1, \ldots, n\}$. Consider the case where the adversary has a decryption key for a set $\{1, 2, 3\}$ (say K_{123}), one for $\{1, 2\}$ (say K_{12}) and a ciphertext for index 3. Then the adversary cannot compute the secret key for

these decryption keys, i.e., $K_{123} = \bigoplus_{\iota=1}^{3} k_\iota$ and $K_{12} = \bigoplus_{\iota=1}^{2} k_\iota$. However, the adversary has k_3, which is appended to the ciphertext for index 3, and knows $K_{123} = K_{12} \oplus k_3$. This correlation becomes a obstacle for the reduction. To circumvent this obstacle, we introduce another encrypting method. That is, we iteratively encrypt a decryption key with SKE, making each bit strings k_ι be a secret key. Then such a correlation does not appear over every decryption key.

The final difficulty is that the adversary asks for decryption keys and ciphertexts in *adaptive manner*. Consequently, the challenger cannot know which type a queried decryption key will be, one that has absent indices or one does not, at the point where the decryption key is queried. Then we need to carefully construct reduction algorithms and evaluate successful probabilities of the reductions.

Concurrent Work

Concurrently and independently to our work, Abdalla et al. [2] have also considered the problem of constructing function-private MIFE scheme for the multi-input inner product functionality supporting a polynomial number of encryption slots under standard assumption. They have first presented a semi-generic scheme that achieves the full-hiding security only in a selective sense. They have subsequently overcome the selective restriction in a concrete instantiation of their semi-generic construction. However, similar to our first MIFE scheme, their construction can only support an apriori fixed number of encryption slots and a fixed slot index set for the multi-input inner product functions. Their concrete adaptively full-hiding MIFE scheme is built in prime order bilinear group setting under the k-MDDH assumption, which subsumes the k-LIN assumption used in our construction. When instantiated under the SXDH assumption, while our construction contains $4n(m^2 - 1)$ more field elements in the master secret key, it involves 2 and $2n + 1$ less group elements in ciphertexts and decryption keys respectively compared to their scheme. On the other hand, our scheme incurs 2 and $2n + 1$ less exponentiations in encryption and key generation procedures respectively, as well as requires $2n$ less pairing operations during decryption compared to theirs. Recall that m and n respectively denote the length of the vectors and the size of the index set associated with the multi-input inner product functionality.

2 Preliminaries

In this section we present various definitions and decisional problems used in this paper.

2.1 Notations

Let $\lambda \in \mathbb{N}$ denotes the security parameter and 1^λ be its unary encoding. Let $\mathbb{N}$ and $\mathbb{Z}$ denote the set of all positive integers and the set of all integers respectively, while $\mathbb{F}_q$, for any prime power $q \in \mathbb{N}$, denotes the finite field of integers modulo p. For $s \in \mathbb{N}$ and $t \in \mathbb{N} \cup \{0\}$ (with $t < s$), we let $[s] = \{1, \ldots, s\}$ and $[t, s] = \{t, \ldots, s\}$. For any set Z, $z \xleftarrow{\mathsf{U}} Z$ represents the process of uniformly sampling

an element z from the set Z, and $|Z|$ signifies the size or cardinality of Z. For a probabilistic algorithm $\mathcal{R}$, we denote by $\varkappa = \mathcal{R}(\Theta; \Phi)$ the output of $\mathcal{R}$ on input Θ and the content of the random tape being Φ, while $\varkappa \xleftarrow{\mathsf{R}} \mathcal{R}(\Theta)$ represents the process of sampling $\varkappa$ from the output distribution of $\mathcal{R}$ on input Θ with a uniform random tape. On the other hand, for any deterministic algorithm $\mathcal{D}$, $\varkappa = \mathcal{D}(\Theta)$ denotes the output of $\mathcal{D}$ on input Θ. We use the abbreviation PPT to mean probabilistic polynomial-time. We assume that all the algorithms are given the unary representation 1^λ of the security parameter λ as input and will not write 1^λ explicitly as input of the algorithms when it is clear from the context. For any finite field $\mathbb{F}_q$ and $m \in \mathbb{N}$, let $\vec{v}$ denotes a vector $(v^{(1)}, \ldots, v^{(m)}) \in \mathbb{Z}^m$ or $\mathbb{F}_q^m$, where $v^{(j)} \in \mathbb{Z}$ or $\mathbb{F}_q$ respectively, for all $j \in [m]$. The all zero vectors in $\mathbb{F}_q^m$ will be denoted by $\vec{0}^m$. For any two vectors $\vec{v}, \vec{w} \in \mathbb{Z}^m$ or $\mathbb{F}_q^m$, $\vec{v} \cdot \vec{w}$ stands for the inner product of the vectors $\vec{v}$ and $\vec{w}$ over the integers, i.e., $\vec{v} \cdot \vec{w} = \sum_{j \in [m]} v^{(j)} w^{(j)} \in \mathbb{Z}$. For any multiplicative cyclic group $\mathbb{G}$ of order q and any generator $g \in \mathbb{G}$, let $\boldsymbol{u}$ represents the m-dimensional vector of group elements $(g^{v^{(1)}}, \ldots, g^{v^{(m)}}) \in \mathbb{G}^m$, for some $\vec{v} \in \mathbb{F}_q^m$. By $\mathbf{1}_{\mathbb{G}}^m$ we denote the m-dimensional vector $(1_{\mathbb{G}}, \ldots, 1_{\mathbb{G}}) \in \mathbb{G}^m$, where $1_{\mathbb{G}}$ represents the identity element of the group $\mathbb{G}$. We use $A = (a_{j,t})$ to represent a matrix with entries $a_{j,t} \in \mathbb{F}_q$. By $A^\intercal$ we will signify the transpose of the matrix A, while by A^* the matrix $(A^{-1})^\intercal$. Let $\mathsf{GL}(\ell, \mathbb{F}_q)$ denotes the set of all $\ell \times \ell$ invertible matrices over $\mathbb{F}_q$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}^+$ is said to be *negligible* if for every $c \in \mathbb{N}$, there exists $T \in \mathbb{N}$ such that for all $\lambda \in \mathbb{N}$ with $\lambda > T$, $|\mathsf{negl}(\lambda)| < 1/\lambda^c$.

2.2 Some Essential Cryptographic Tools

Definition 2.1 (Pseudorandom Functions: PRFs): A pseudorandom function family $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ with key space $\mathcal{K} = \{\mathcal{K}_\lambda\}_{\lambda \in \mathbb{N}}$, domain $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$, and range $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ is a function family that consists of functions $F_\lambda : \mathcal{K}_\lambda \times \mathcal{X}_\lambda \to \mathcal{Y}_\lambda$. Let $\mathcal{R}_\lambda$ be a set of functions consists of all functions whose domain and range are $\mathcal{X}_\lambda$ and $\mathcal{Y}_\lambda$ respectively. For all PPT adversary $\mathcal{A}$, the following condition holds;

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{PRF}}(\lambda) = \left| \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}^{F(K,\cdot)}] - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}^{R(\cdot)}] \right| \leq \mathsf{negl}(\lambda),$$

where $F \in \mathcal{F}_\lambda$, $K \xleftarrow{\mathsf{U}} \mathcal{K}_\lambda$, and $R \xleftarrow{\mathsf{U}} \mathcal{R}_\lambda$.

Definition 2.2 (Symmetric Key Encryption: SKE): A symmetric key encryption consists of a tuple of three PPT algorithms ($\mathsf{SKE.KeyGen}$, $\mathsf{SKE.Encrypt}$, $\mathsf{SKE.Decrypt}$). $\mathsf{SKE.KeyGen}$ takes 1^λ as an input and outputs a secret key K. $\mathsf{SKE.Encrypt}$ takes a secret key K and a message m and outputs a ciphertext c. $\mathsf{SKE.Decrypt}$ takes a secret key K and a ciphertext c and outputs a message m'. Correctness of SKE is that

$\Pr[m = m' | K \xleftarrow{\mathsf{R}} \mathsf{SKE.KeyGen}, m' = \mathsf{SKE.Decrypt}(K, \mathsf{SKE.Encrypt}(K, m))] = 1.$

A semantically secure symmetric key encryption scheme satisfies the following condition. For all PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathrm{SKE}}(\lambda) = \left| \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{O}_0(\cdot)}] - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}^{\mathcal{O}_1(\cdot)}] \right| \leq \mathsf{negl}(\lambda),$$

where an oracle $\mathcal{O}_{\beta\in\{0,1\}}$ chooses a random secret key K as $K \xleftarrow{\mathsf{R}} \mathsf{SKE.KeyGen}$ and when it takes a pair of messages (m_0, m_1), it returns $\mathsf{SKE.Encrypt}(K, m_\beta)$.

2.3 Bilinear Groups and Dual Pairing Vector Spaces

Definition 2.3 (Bilinear Group): A bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime integer $q \in \mathbb{N}$; cyclic multiplicative groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ of order q each with polynomial-time computable group operations; generators $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$; and a polynomial-time computable non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, i.e., e satisfies the following two properties:

- *Bilinearity*: $e(g_1^{\zeta}, g_2^{\eta}) = e(g_1, g_2)^{\zeta\eta}$ for all $\zeta, \eta \in \mathbb{F}_q$.
- *Non-degeneracy*: $e(g_1, g_2) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ denotes the identity element of the group $\mathbb{G}_T$.

Definition 2.4 (Dual Pairing Vector Spaces: DPVS [34, 35]): A dual pairing vector space (DPVS) $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e)$ by the direct product of a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$ is a tuple of a prime integer q; m-dimensional vector spaces $\mathbb{V}_\chi = \mathbb{G}_\chi^m$ over $\mathbb{F}_q$, for $\chi \in [2]$, under vector addition '$\boxplus$' and scalar multiplication '$\circ$' defined componentwise; canonical bases $\mathbb{A}_\chi = \{\boldsymbol{a}_{\chi,j} = (\overbrace{1_{\mathbb{G}_\chi}, \ldots, 1_{\mathbb{G}_\chi}}^{j-1}, g_\chi, \overbrace{1_{\mathbb{G}_\chi}, \ldots, 1_{\mathbb{G}_\chi}}^{m-j})\}_{j\in[m]}$ of $\mathbb{V}_\chi$, for $\chi \in [2]$, where $1_{\mathbb{G}_\chi}$ is the identity element of the group $\mathbb{G}_\chi$, for $\chi \in [2]$; and a pairing $e : \mathbb{V}_1 \times \mathbb{V}_2 \to \mathbb{G}_T$ defined by $e(\boldsymbol{v}, \boldsymbol{w}) = \prod_{j\in[m]} e(g_1^{v^{(j)}}, g_2^{w^{(j)}}) \in \mathbb{G}_T$, for all $\boldsymbol{v} = (g_1^{v^{(1)}}, \ldots, g_1^{v^{(q)}}) \in \mathbb{V}_1$, $\boldsymbol{w} = (g_2^{w^{(1)}}, \ldots, g_2^{w^{(q)}}) \in \mathbb{V}_2$. Observe that the newly defined map e is also non-degenerate bilinear, i.e., e satisfies the following two properties:

- *Bilinearity*: $e(\mu \circ \boldsymbol{v}, \eta \circ \boldsymbol{w}) = e(\boldsymbol{v}, \boldsymbol{w})^{\mu\eta}$, for $\mu, \eta \in \mathbb{F}_q$, $\boldsymbol{v} \in \mathbb{V}_1$, and $\boldsymbol{w} \in \mathbb{V}_2$.
- *Non-degeneracy*: If $e(\boldsymbol{v}, \boldsymbol{w}) = 1_{\mathbb{G}_T}$ for all $\boldsymbol{w} \in \mathbb{V}_2$, then $\boldsymbol{v} = \boldsymbol{1}_{\mathbb{G}_1}^m$.

We will often omit the symbol '$\circ$' for scalar multiplication and abuse '$+$' for the vector addition '$\boxplus$' when it is clear from the context. For any set $\mathbb{W} = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_m\}$ of vectors in $\mathbb{V}_\chi$, for $\chi \in [2]$, and any vector $\vec{v} \in \mathbb{F}_q^m$, let $(\vec{v})_{\mathbb{W}}$ represents the vector in $\mathbb{V}_\chi$ formed by the linear combination of the members of $\mathbb{W}$ with the entries of $\vec{v}$ as the coefficients, i.e., $(\vec{v})_{\mathbb{W}} = \sum_{j\in[m]} v^{(j)} \boldsymbol{w}_j \in \mathbb{V}_\chi$. Also, for any vector $\boldsymbol{v} \in \mathbb{V}_\chi$, for $\chi \in [2]$, and any matrix $A = (a_{j,t})$ with entries $a_{j,t} \in \mathbb{F}_q$, for $j, t \in [m]$, we denote by $\boldsymbol{v}A$ the m-dimensional vector $(g_\chi^{\sum_{j\in[m]} a_{j,1} v^{(j)}}, \ldots, g_\chi^{\sum_{j\in[m]} a_{j,m} v^{(j)}}) \in \mathbb{V}_\chi$. The DPVS generation algorithm $\mathcal{G}_{\mathrm{DPVS}}$ takes input the unary encoded security parameter 1^λ, a dimension value $m \in \mathbb{N}$, along with a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{BPG}}()$, and outputs a description $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e)$ of DPVS with m-dimensional $\mathbb{V}_1$ and $\mathbb{V}_2$.

We now describe random *dual orthogonal* basis generator $\mathcal{G}_{\mathrm{OB}}$ [34,35] in Fig. 2.1. This algorithm would be utilized as a sub-routine in our constructions.

$\mathcal{G}_{\text{OB}}(m, \mathsf{params}_{\mathbb{V}}, \nu)$: This algorithm takes as input the unary encoded security parameter 1^{λ}, a dimension value $m \in \mathbb{N}$, a m-dimensional DPVS $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{R} \mathcal{G}_{\text{DPVS}}(m, \mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e))$, along with a random value $\nu \xleftarrow{U} \mathbb{F}_q \backslash \{0\}$, and performs the following operations:

1. It first samples random $B = (b_{j,t}) \xleftarrow{U} \mathsf{GL}(m, \mathbb{F}_q)$.
2. Next, it computes $B^{\bigstar} = (b^*_{j,t}) = \nu B^*$.
3. For all $j \in [m]$, let $\vec{b}_j$ and $\vec{b}^*_j$ represent the j^{th} row-vectors of B and $B^{\bigstar}$ respectively. It computes $\boldsymbol{b}_j = (\vec{b}_j)_{\mathbb{A}_1}, \boldsymbol{b}^*_j = (\vec{b}^*_j)_{\mathbb{A}_2}$, for $j \in [m]$, and sets
$$\mathbb{B} = \{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_m\}, \mathbb{B}^* = \{\boldsymbol{b}^*_1, \ldots, \boldsymbol{b}^*_m\}.$$
Clearly $\mathbb{B}$ and $\mathbb{B}^*$ form bases of the vector spaces $\mathbb{V}_1 = \mathbb{G}_1^m$ and $\mathbb{V}_2 = \mathbb{G}_2^m$ respectively. Also, note that $\mathbb{B}$ and $\mathbb{B}^*$ are dual orthogonal in the sense that for all $j, j' \in [m]$,
$$e(\boldsymbol{b}_j, \boldsymbol{b}^*_{j'}) = \begin{cases} g_T, & \text{if } j = j' \\ 1_{\mathbb{G}_T}, & \text{otherwise} \end{cases},$$
where $g_T = e(g_1, g_2)^{\nu}$.
4. It returns $(\mathbb{B}, \mathbb{B}^*)$.

Fig. 2.1: Dual Orthogonal Basis Generator $\mathcal{G}_{\text{OB}}$

2.4 Complexity Assumptions

Assumption 1 (k-Linear: k-LIN [37]): Fix a number $\chi \in [2]$. The k-LIN problem is to guess a bit $\hat{\beta} \xleftarrow{U} \{0,1\}$ given $\varepsilon_{\hat{\beta}} = (\mathsf{params}_{\mathbb{G}}, g_{\chi}^{\xi_1}, \ldots, g_{\chi}^{\xi_k}, g_{\chi}^{\delta_1 \xi_1}, \ldots, g_{\chi}^{\delta_k \xi_k}, \Re_{\hat{\beta}})$; where $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{R} \mathcal{G}_{\text{BPG}}()$; $\xi_1, \ldots, \xi_k, \sigma \xleftarrow{U} \mathbb{F}_q \backslash \{0\}$; $\delta_1, \ldots, \delta_k \xleftarrow{U} \mathbb{F}_q$; and $\Re_{\hat{\beta}} = g_{\chi}^{\sum_{j \in [k]} \delta_j}$ or $g_{\chi}^{\sigma + \sum_{j \in [k]} \delta_j}$ according as $\hat{\beta} = 0$ or 1. The k-LIN assumption states that for any PPT algorithm $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{F}$ in deciding the k-LIN problem,
$$\mathsf{Adv}^{k\text{-LIN}}_{\mathcal{A}}(\lambda) = \left| \mathsf{Pr}[1 \xleftarrow{R} \mathcal{A}(\varepsilon_0)] - \mathsf{Pr}[1 \xleftarrow{R} \mathcal{A}(\varepsilon_1)] \right| \leq \mathsf{negl}(\lambda),$$
for some negligible function negl.

We now define a set of decisional problems. We rely on the hardness of these problems for deriving security of our constructions. We justify the reducibility of the hardness of these problems to that of the k-LIN problem in the full version of this paper.

Definition 2.5 (Problem 1): Problem 1 is to guess a bit $\hat{\beta} \xleftarrow{U} \{0,1\}$ given $\varrho_{\hat{\beta}} = (\mathsf{params}_{\mathbb{V}}, g_T, \{\widehat{\mathbb{B}}_{\iota}, \widehat{\mathbb{B}}^*_{\iota}\}_{\iota \in [n]}, \{\boldsymbol{\Upsilon}_{\iota, \hat{\beta}}\}_{\iota \in [n]})$; where $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{R} \mathcal{G}_{\text{BPG}}()$; $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{R} \mathcal{G}_{\text{DPVS}}(2m + 2k + 1, \mathsf{params}_{\mathbb{G}})$; $\nu \xleftarrow{U} \mathbb{F}_q \backslash \{0\}$; $g_T = e(g_1, g_2)^{\nu}$; $(\mathbb{B}_{\iota}, \mathbb{B}^*_{\iota}) \xleftarrow{R} \mathcal{G}_{\text{OB}}(2m+2k+1, \mathsf{params}_{\mathbb{V}}, \nu)$; $\widehat{\mathbb{B}}_{\iota} = \{\boldsymbol{b}_{\iota,1}, \ldots, \boldsymbol{b}_{\iota,2m+1}, \boldsymbol{b}_{\iota,2m+k+1}, \ldots, \boldsymbol{b}_{\iota,2m+2k}\}$, $\widehat{\mathbb{B}}^*_{\iota} = \{\boldsymbol{b}^*_{\iota,1}, \ldots, \boldsymbol{b}^*_{\iota,2m+k}\}$, for $\iota \in [n]$; $\alpha_1, \ldots, \alpha_k \xleftarrow{U} \mathbb{F}_q$; $\Im \xleftarrow{U} \mathbb{F}_q \backslash \{0\}$; and $\boldsymbol{\Upsilon}_{\iota, \hat{\beta}} = (\vec{0}^{2m+k}, \alpha_1, \ldots, \alpha_k, 0)_{\mathbb{B}_{\iota}}$ or $(\vec{0}^{2m+k}, \alpha_1, \ldots, \alpha_k, \Im)_{\mathbb{B}_{\iota}}$ according as $\hat{\beta} = 0$ or 1. For any PPT algorithm $\mathcal{A}$, the

advantage of $\mathcal{A}$ in deciding Problem 1 is defined as

$$\mathsf{Adv}^{\mathrm{P1}}_{\mathcal{A}}(\lambda) = \left|\mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}(\varrho_0)] - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}(\varrho_1)]\right|.$$

Definition 2.6 (Problem 1*): Problem 1* is to guess a bit $\hat{\beta} \xleftarrow{\mathsf{U}} \{0,1\}$ given $\varrho_{\hat{\beta}} = (\mathsf{params}_{\mathbb{V}}, g_T, \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota\in[n]}, \{\boldsymbol{\Upsilon}_{\iota,\hat{\beta}}\}_{\iota\in[n]})$; where $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{BPG}}()$; $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{DPVS}}(2m+2k+1, \mathsf{params}_{\mathbb{G}})$; $\nu \xleftarrow{\mathsf{U}} \mathbb{F}_q\backslash\{0\}$; $g_T = e(g_1, g_2)^\nu$; $(\mathbb{B}_\iota, \mathbb{B}^*_\iota) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{OB}}(2m+2k+1, \mathsf{params}_{\mathbb{V}}, \nu)$, for $\iota \in [n]$; $\widehat{\mathbb{B}}_\iota = \{\boldsymbol{b}_{\iota,1}, \ldots, \boldsymbol{b}_{\iota,2m+1}, \boldsymbol{b}_{\iota,2m+k+1}, \ldots, \boldsymbol{b}_{\iota,2m+2k}\}$, $\widehat{\mathbb{B}}^*_\iota = \{\boldsymbol{b}^*_{\iota,1}, \ldots, \boldsymbol{b}^*_{\iota,2m+k}, \boldsymbol{b}^*_{\iota,2m+2k+1}\}$, for $\iota \in [n]$; $\alpha_1, \ldots, \alpha_k \xleftarrow{\mathsf{U}} \mathbb{F}_q$; $\Im \xleftarrow{\mathsf{U}} \mathbb{F}_q\backslash\{0\}$; and $\boldsymbol{\Upsilon}_{\iota,\hat{\beta}} = (\vec{0}^{2m}, \alpha_1, \ldots, \alpha_k, \vec{0}^k, 0)_{\mathbb{B}^*_\iota}$ or $(\vec{0}^{2m}, \alpha_1, \ldots, \alpha_k, \vec{0}^k, \Im)_{\mathbb{B}^*_\iota}$ according as $\hat{\beta} = 0$ or 1, for $\iota \in [n]$. For any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in deciding Problem 1* is defined as

$$\mathsf{Adv}^{\mathrm{P1}^*}_{\mathcal{A}}(\lambda) = \left|\mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}(\varrho_0)] - \mathsf{Pr}[1 \xleftarrow{\mathsf{R}} \mathcal{A}(\varrho_1)]\right|.$$

2.5 Notion of Full-Hiding Multi-Input Inner Product Functional Encryption

Definition 2.7 (Multi-Input Inner Product Functionality): An unbounded-arity multi-input inner product function family $\mathcal{F}^{m,\mathscr{B}}_\lambda = \{\mathcal{F}^{m,\mathscr{B}}_S\}$, for some $m, \mathscr{B} \in \mathbb{N}$, consists of the sub-families $\mathcal{F}^{m,\mathscr{B}}_S$ of bounded-arity multi-input inner product functions, where each subfamily $\mathcal{F}^{m,\mathscr{B}}_S$ is parameterized with an index set $S \subseteq [t(\lambda)]$ for any polynomial t, and contains functions $f_{\{\vec{y}_\iota\}_{\iota\in S}} : (\mathbb{Z}^m)^{|S|} \to \mathbb{Z}$ associated with sets of vectors $\{\vec{y}_\iota\}_{\iota\in S}$ such that each vector $\vec{y}_\iota \in \mathbb{Z}^m$, where $f_{\{\vec{y}_\iota\}_{\iota\in S}}(\{\vec{x}_\iota\}_{\iota\in S}) = \sum_{\iota\in S} \vec{x}_\iota \cdot \vec{y}_\iota$, for all sets of vectors $\{\vec{x}_\iota\}_{\iota\in S}$ such that each vector $\vec{x}_\iota \in \mathbb{Z}^m$ and the norm of the inner product $|\vec{x}_\iota \cdot \vec{y}_\iota| \leq \mathscr{B}$ for all $\iota \in S$.

Without loss of generality, when dealing with MIFE for some bounded-arity multi-input inner product function family $\mathcal{F}^{m,\mathscr{B}}_S$, we consider the associated index set S to be $[n]$, and denote the function family as $\mathcal{F}^{m,\mathscr{B}}_n$, where $n = |S|$.

Definition 2.8 (Full-Hiding Private Key Bounded-Arity Multi-Input Inner Product Functional Encryption: FH-MIPE): A full-hiding private key bounded-arity multi-input inner product functional encryption scheme for an inner product function family $\mathcal{F}^{m,\mathscr{B}}_n$ consists of the following polynomial-time algorithms:

FH-MIPE.Setup$(m, n, \mathscr{B})$: This algorithm takes as input the unary encoded security parameter 1^λ, along with the length $m \in \mathbb{N}$ of vectors, the arity $n \in \mathbb{N}$ of the multi-input inner product functionality, and the bound $\mathscr{B} \in \mathbb{N}$ on the size of each component inner products. It generates a master secret key MSK and the corresponding public parameters PP. Observe that we are considering private key setting and hence PP is not sufficient to encrypt. It merely includes some public informations required for decryption, e.g., the group description in a bilinear-map-based construction.

FH-MIPE.KeyGen$(\mathrm{PP}, \mathrm{MSK}, \{\vec{y}_\iota\}_{\iota\in[n]})$: On input the public parameters PP, the master secret key MSK, along with a set of n vectors $\{\vec{y}_\iota\}_{\iota\in[n]}$ such that $\vec{y}_\iota \in \mathbb{Z}^m$ for all $\iota \in [n]$, this algorithm outputs a decryption key SK.

FH-MIPE.Encrypt$(\mathrm{PP}, \mathrm{MSK}, \iota, \vec{x}_\iota)$: This algorithm upon input the public parameters PP, the master secret key MSK, an index $\iota \in [n]$, and a vector $\vec{x}_\iota \in \mathbb{Z}_p^m$, outputs a ciphertext CT_ι, which includes the index ι in the clear.

FH-MIPE.Decrypt$(\mathrm{PP}, \mathrm{SK}, \{\mathrm{CT}_\iota\}_{\iota\in[n]})$: On input the public parameters PP, a decryption key SK, along with a set of n ciphertexts $\{\mathrm{CT}_\iota\}_{\iota\in[n]}$, where for all $\iota \in [n]$, CT_ι is a ciphertext prepared for the ι^{th} index, this algorithm either outputs a value $\Lambda \in \mathbb{Z}$ or the distinguished symbol $\perp$ indicating failure.

The algorithm FH-MIPE.Decrypt is deterministic while all the others are probabilistic. The algorithms satisfy the following correctness and security requirements.

■ **Correctness**: An FH-MIPE scheme is correct if for any security parameter $\lambda \in \mathbb{N}$, any polynomial n in λ, any $m, \mathscr{B} \in \mathbb{N}$, any two sets of n vectors $\{\vec{x}_\iota\}_{\iota\in[n]}, \{\vec{y}_\iota\}_{\iota\in[n]}$ such that $\vec{x}_\iota, \vec{y}_\iota \in \mathbb{Z}^m$ with $|\vec{x}_\iota \cdot \vec{y}_\iota| \leq \mathscr{B}$ for all $\iota \in [n]$, we have

$$\mathsf{Pr}\big[\mathsf{FH\text{-}MIPE.Decrypt}(\mathrm{PP}, \mathrm{SK}, \{\mathrm{CT}_\iota\}_{\iota\in[n]}) = \sum_{\iota\in[n]} \vec{x}_\iota \cdot \vec{y}_\iota :$$

$$(\mathrm{PP}, \mathrm{MSK}) \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}MIPE.Setup}(m, n, \mathscr{B});$$

$$\mathrm{SK} \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}MIPE.KeyGen}(\mathrm{PP}, \mathrm{MSK}, \{\vec{y}_\iota\}_{\iota\in[n]});$$

$$\{\mathrm{CT}_\iota \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}MIPE.Encrypt}(\mathrm{PP}, \mathrm{MSK}, \iota, \vec{x}_\iota)\}_{\iota\in[n]}\big] \geq 1 - \mathsf{negl}(\lambda),$$

for some negligible function negl.

■ **Full-Hiding Security**: The (indistinguishability-based) full-hiding security notion for a private key bounded-arity FH-MIPE scheme is formalized through the experiment $\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(\beta)$, for random $\beta \xleftarrow{\mathsf{U}} \{0,1\}$, which involves a PPT adversary $\mathcal{A}$ and a PPT challenger $\mathcal{B}$. The experiment is described below:

Setup: $\mathcal{B}$ generates $(\mathrm{PP}, \mathrm{MSK}) \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}MIPE.Setup}(m, n, \mathscr{B})$ and provides PP to $\mathcal{A}$.

Query Phase: $\mathcal{A}$ is allowed to adaptively make any polynomial number of queries of the following two types in arbitrary order:

- *Decryption key query*: In response to the i^{th} decryption key query of $\mathcal{A}$ corresponding to a pair of sets of vectors $(\{\vec{y}_{\iota,i,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,i,1}\}_{\iota\in[n]})$ such that $\vec{y}_{\iota,i,0}, \vec{y}_{\iota,i,1} \in \mathbb{Z}^m$ for all $\iota \in [n]$, $\mathcal{B}$ forms a decryption key $\mathrm{SK}_i^* \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}MIPE.KeyGen}(\mathrm{PP}, \mathrm{MSK}, \{\vec{y}_{\iota,i,\beta}\}_{\iota\in[n]})$ and hands SK_i^* to $\mathcal{A}$.
- *Ciphertext query*: To answer a ciphertext query of $\mathcal{A}$ for the ι^{th} index corresponding to a pair of vectors $(\vec{x}_{\iota,t_\iota,0}, \vec{x}_{\iota,t_\iota,1}) \in (\mathbb{Z}^m)^2$, $\mathcal{B}$ prepares a ciphertext $\mathrm{CT}_{\iota,t_\iota}^* \xleftarrow{\mathsf{R}} \mathsf{FH.MIPE.Encrypt}(\mathrm{PP}, \mathrm{MSK}, \vec{x}_{\iota,t_\iota,\beta})$ and gives $\mathrm{CT}_{\iota,t_\iota}^*$ to $\mathcal{A}$.

Let the total number of decryption key query made by $\mathcal{A}$ be $q_{\mathrm{KEY}}(\geq 0)$ and the total number of ciphertext query made for the ι^{th} index be $q_{\mathrm{CT},\iota}(\geq 0)$.

The restrictions on the queries of $\mathcal{A}$ are that if $q_{\mathrm{CT},\iota} \geq 1$ for all $\iota \in [n]$, then for all $i \in [q_{\mathrm{KEY}}]$ and for all $(t_1, \ldots, t_n) \in [q_{\mathrm{CT},1}] \times \ldots \times [q_{\mathrm{CT},n}]$, we must have

$$\sum_{\iota \in [n]} \vec{x}_{\iota,t_\iota,0} \cdot \vec{y}_{\iota,i,0} = \sum_{\iota \in [n]} \vec{x}_{\iota,t_\iota,1} \cdot \vec{y}_{\iota,i,1}. \tag{2.1}$$

Guess: $\mathcal{A}$ eventually outputs a guess bit $\beta' \in \{0,1\}$, which is the output of the experiment.

A private key FH-MIPE scheme is said to be full-hiding if for any PPT adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment,

$$\mathsf{Adv}^{\text{FH-MIPE}}_{\mathcal{A}}(\lambda) = \left|\mathsf{Pr}[\mathsf{Expt}^{\text{FH-MIPE}}_{\mathcal{A}}(0) = 1] - \mathsf{Pr}[\mathsf{Expt}^{\text{FH-MIPE}}_{\mathcal{A}}(1) = 1]\right| \leq \mathsf{negl}(\lambda),$$

for some negligible function negl.

Definition 2.9 (Full-Hiding Unbounded Private Key Multi-Input Inner Product Functional Encryption: FH-UMIPE): An unbounded full-hiding private key multi-input inner product functional encryption scheme for an inner product function family $\mathcal{F}^{m,\mathscr{B}}_{\lambda}$ consists of the following polynomial-time algorithms:

- $\mathsf{FH\text{-}UMIPE.Setup}(m, \mathscr{B})$: This algorithm takes as input the unary encoded security parameter 1^λ, along with the length $m \in \mathbb{N}$ of vectors, and the bound $\mathscr{B} \in \mathbb{N}$ of each inner product values. It generates a master secret key MSK and the corresponding public parameters PP. It publishes PP, while keeps MSK to itself.
- $\mathsf{FH\text{-}UMIPE.KeyGen}(\mathrm{PP}, \mathrm{MSK}, S, \{\vec{y}_\iota\}_{\iota \in S})$: On input the public parameters PP, the master secret key MSK, a set of indices $S \subseteq [t(\lambda)]$ where t is any polynomial, along with an $|S|$-tuple of vectors $\{\vec{y}_\iota\}_{\iota \in S} \in (\mathbb{Z}^m)^{|S|}$, this algorithm provides a decryption key SK_S including the set S explicitly.
- $\mathsf{FH\text{-}UMIPE.Encrypt}(\mathrm{PP}, \mathrm{MSK}, \iota, \vec{x}_\iota)$: On input the public parameters PP, the master secret key MSK, an index $\iota \in [2^\lambda]$, and a vector $\vec{x}_\iota \in \mathbb{Z}^m$, outputs a ciphertext CT_ι, which includes the index ι in the clear.
- $\mathsf{FH\text{-}UMIPE.Decrypt}(\mathrm{PP}, \mathrm{SK}_S, \{\mathrm{CT}_\iota\}_{\iota \in S})$: On input the public parameters PP, a decryption key SK_S associated with S, along with a tuple of $|S|$ ciphertexts $\{\mathrm{CT}_\iota\}_{\iota \in S}$, where CT_ι is a ciphertext prepared for the index ι, a decrypter either outputs a value $\Lambda \in \mathbb{N}$ or the distinguished symbol $\perp$ indicating failure.

The algorithm FH-UMIPE.Decrypt is deterministic while all the others are probabilistic. The algorithms satisfy the following correctness and security requirements.

■ **Correctness**: An FH-UMIPE scheme is correct if for any $m, \mathscr{B}, \lambda \in \mathbb{N}$, any set of indices $S \subseteq [t(\lambda)]$, where t is any polynomial, any two $|S|$-tuples of vectors $\{\vec{x}_\iota\}_{\iota \in S}, \{\vec{y}_\iota\}_{\iota \in S} \in (\mathbb{Z}^m)^{|S|}$ with $|\vec{x}_\iota \cdot \vec{y}_\iota| \leq \mathscr{B}$ for all $\iota \in S$, we have

$$\mathsf{Pr}\Big[\mathsf{FH\text{-}UMIPE.Decrypt}(\mathrm{PP}, \mathrm{SK}_S, \{\mathrm{CT}_\iota\}_{\iota \in S}) = \sum_{\iota \in S} \vec{x}_\iota \cdot \vec{y}_\iota :$$

$$(\mathrm{PP}, \mathrm{MSK}) \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}UMIPE.Setup}(m, \mathscr{B});$$

$$\mathrm{SK}_S \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}UMIPE.KeyGen}(\mathrm{PP}, \mathrm{MSK}, S, \{\vec{y}_\iota\}_{\iota \in S});$$

$$\{\mathrm{CT}_\iota \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}UMIPE.Encrypt}(\mathrm{PP}, \iota, \vec{x}_\iota)\}_{\iota \in S}\Big] \geq 1 - \mathsf{negl}(\lambda)$$

■ **Full-Hiding Security**: The (indistinguishability-based) full-hiding security notion for a private key FH-UMIPE scheme is formalized through the experiment $\mathsf{Expt}_{\mathcal{A}}^{\text{FH-UMIPE}}(\beta)$, for random $\beta \xleftarrow{\mathsf{U}} \{0,1\}$, which involves a PPT adversary $\mathcal{A}$ and a PPT challenger $\mathcal{B}$. The experiment is described below:

Setup: $\mathcal{B}$ generates $(\text{PP}, \text{MSK}) \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}UMIPE.Setup}(m, \mathscr{B})$ and gives PP to $\mathcal{A}$.

Query Phase: $\mathcal{A}$ is allowed to adaptively make any polynomial number of queries of the following two types in arbitrary order:

- *Decryption key query*: In response to the i^{th} decryption key query of $\mathcal{A}$ corresponding to a set of indices $S_i \subseteq [t(\lambda)]$ for any polynomial t and a pair of $|S_i|$-tuples of vectors $\{\vec{y}_{\iota,i,0}, \vec{y}_{\iota,i,1}\}_{\iota \in S_i} \in ((\mathbb{Z}^m)^{|S_i|})^2$, $\mathcal{B}$ forms a decryption key $\text{SK}^*_{S_i,i} \xleftarrow{\mathsf{R}} \mathsf{FH\text{-}UMIPE.KeyGen}(\text{PP}, \text{MSK}, S_i, \{\vec{y}_{\iota,i,\beta}\}_{\iota \in S_i})$ and hands $\text{SK}^*_{S_i,i}$ to $\mathcal{A}$.
- *Ciphertext query*: To answer a ciphertext query of $\mathcal{A}$ for the ι^{th} index corresponding to a pair of vectors $(\vec{x}_{\iota,t_\iota,0}, \vec{x}_{\iota,t_\iota,1}) \in (\mathbb{Z}^m)^2$, $\mathcal{B}$ prepares a ciphertext $\text{CT}^*_{\iota,t_\iota} \xleftarrow{\mathsf{R}} \mathsf{UFH.MIPE.Encrypt}(\text{PP}, \text{MSK}, \vec{x}_{\iota,t_\iota,\beta})$ and gives $\text{CT}^*_{\iota,t_\iota}$ to $\mathcal{A}$.

Let the total number of decryption key query made by $\mathcal{A}$ be $q_{\text{KEY}}(\geq 0)$ and the total number of ciphertext query made for the ι^{th} index be $q_{\text{CT},\iota}(\geq 0)$. The restrictions on the queries of $\mathcal{A}$ are that for each $i \in [q_{\text{KEY}}]$, if $q_{\text{CT},\iota} \geq 1$ for all $\iota \in S_i$, then for all $\{t_\iota\}_{\iota \in S_i} \in \prod_{\iota \in S_i} [q_{\text{CT},\iota}]$ we must have $\sum_{\iota \in S_i} \vec{x}_{\iota,t_\iota,0} \cdot \vec{y}_{\iota,i,0} = \sum_{\iota \in S_i} \vec{x}_{\iota,t_\iota,1} \cdot \vec{y}_{\iota,i,1}$.

Guess: $\mathcal{A}$ eventually outputs a guess bit $\beta' \in \{0,1\}$, which is the output of the experiment.

A private key FH-UMIPE scheme is said to be full-hiding if for any PPT adversary $\mathcal{A}$, for any security parameter λ, the advantage of $\mathcal{A}$ in the above experiment,
$\mathsf{Adv}_{\mathcal{A}}^{\text{FH-UMIPE}}(\lambda) = \left|\mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\text{FH-UMIPE}}(0) = 1] - \mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\text{FH-UMIPE}}(1) = 1]\right| \leq \mathsf{negl}(\lambda)$,
for some negligible function negl.

3 The Proposed Full-Hiding Bounded Multi-Input Inner Product Functional Encryption Scheme

In this section, we present our FH-MIPE scheme.

3.1 Construction

$\mathsf{FH\text{-}MIPE.Setup}(m, n, \mathscr{B})$: This algorithm takes as input the unary encoded security parameter 1^λ, the length $m \in \mathbb{N}$ of vectors, the arity $n \in \mathbb{N}$ of the multi-input inner product functionality, and the bound $\mathscr{B} \in \mathbb{N}$ on each component inner product. It proceeds as follows:

1. First, it generates a bilinear group $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\text{BPG}}()$ with $q \gg n\mathscr{B}$.
2. Next, it creates $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\text{DPVS}}(2m + 2k + 1, \mathsf{params}_{\mathbb{G}})$.

3. Then, it samples random $\nu \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$, and computes $g_T = e(g_1, g_2)^{\nu}$.
4. After that, for $\iota \in [n]$, it generates $(\mathbb{B}_\iota = \{\boldsymbol{b}_{\iota,1}, \ldots, \boldsymbol{b}_{\iota,2m+2k+1}\}, \mathbb{B}^*_\iota = \{\boldsymbol{b}^*_{\iota,1}, \ldots, \boldsymbol{b}^*_{\iota,2m+2k+1}\}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{OB}}(2m+2k+1, \mathsf{params}_{\mathbb{V}}, \nu)$ and sets
$$\widehat{\mathbb{B}}_\iota = \{\boldsymbol{b}_{\iota,1}, \ldots, \boldsymbol{b}_{\iota,m}, \boldsymbol{b}_{\iota,2m+1}, \boldsymbol{b}_{\iota,2m+k+1}, \ldots, \boldsymbol{b}_{\iota,2m+2k}\},$$
$$\widehat{\mathbb{B}}^*_\iota = \{\boldsymbol{b}^*_{\iota,1}, \ldots, \boldsymbol{b}^*_{\iota,m}, \boldsymbol{b}^*_{\iota,2m+1}, \ldots, \boldsymbol{b}^*_{\iota,2m+k}\}.$$
5. It publishes the public parameters $\mathrm{PP} = (\mathsf{params}_{\mathbb{V}}, g_T)$, while sets the master secret key $\mathrm{MSK} = \{\widehat{\mathbb{B}}_\iota, \widehat{\mathbb{B}}^*_\iota\}_{\iota \in [n]}$.

$\mathsf{FH\text{-}MIPE.KeyGen}(\mathrm{PP}, \mathrm{MSK}, \{\vec{y}_\iota\}_{\iota \in [n]})$: On input the public parameters PP, the master secret key MSK, along with a set of n vectors $\{\vec{y}_\iota\}_{\iota \in [n]}$ such that $\vec{y}_\iota \in \mathbb{F}_q^m$, this algorithm executes the following steps:

1. First, it samples random $r_\iota, \gamma_{\iota,1}, \ldots, \gamma_{\iota,k-1} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, for $\iota \in [n]$, subject to the restriction that $\sum_{\iota \in [n]} r_\iota = 0$.
2. Next, for each $\iota \in [n]$, it computes
$$\begin{aligned}\boldsymbol{k}_\iota &= \sum_{j \in [m]} y^{(j)}_\iota \boldsymbol{b}^*_{\iota,j} + r_\iota \boldsymbol{b}^*_{\iota,2m+1} + \sum_{j \in [k-1]} \gamma_{\iota,j} \boldsymbol{b}^*_{\iota,2m+1+j} \\ &= (\vec{y}_\iota, \vec{0}^m, r_\iota, \gamma_{\iota,1}, \ldots, \gamma_{\iota,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_\iota},\end{aligned}$$
by making use of $\widehat{\mathbb{B}}^*_\iota$ extracted from MSK.
3. It outputs the decryption key $\mathrm{SK} = \{\boldsymbol{k}_\iota\}_{\iota \in [n]}$.

$\mathsf{FH\text{-}MIPE.Encrypt}(\mathrm{PP}, \mathrm{MSK}, \iota, \vec{x}_\iota)$: Taking as input the public parameters PP, the master secret key MSK, an index $\iota \in [n]$, along with a vector $\vec{x}_\iota \in \mathbb{F}_q^m$, this algorithm performs the following steps:

1. It selects random $\varphi_{\iota,1}, \ldots, \varphi_{\iota,k} \xleftarrow{\mathsf{U}} \mathbb{F}_q$, and computes
$$\begin{aligned}\boldsymbol{c}_\iota &= \sum_{j \in [m]} x^{(j)}_\iota \boldsymbol{b}_{\iota,j} + \boldsymbol{b}_{\iota,2m+1} + \sum_{j \in [k]} \varphi_{\iota,j} \boldsymbol{b}_{\iota,2m+k+j} \\ &= (\vec{x}_\iota, \vec{0}^m, 1, \vec{0}^{k-1}, \varphi_{\iota,1}, \ldots, \varphi_{\iota,k}, 0)_{\mathbb{B}_\iota},\end{aligned}$$
by utilizing $\widehat{\mathbb{B}}_\iota$ extracted from MSK.
2. It outputs the ciphertext $\mathrm{CT}_\iota = (\iota, \boldsymbol{c}_\iota)$.

$\mathsf{FH\text{-}MIPE.Decrypt}(\mathrm{PP}, \mathrm{SK}, \{\mathrm{CT}_\iota\}_{\iota \in [n]})$: This algorithm takes as input the public parameters PP, a decryption key $\mathrm{SK} = \{\boldsymbol{k}_\iota\}_{\iota \in [n]}$, and a set of n ciphertexts $\{\mathrm{CT}_\iota = (\iota, \boldsymbol{c}_\iota)\}_{\iota \in [n]}$. It does the following:

1. It first computes $L_T = \prod_{\iota \in [n]} e(\boldsymbol{c}_\iota, \boldsymbol{k}_\iota)$.
2. Then, it attempts to determine a value $\Lambda \in \mathbb{Z}$ such that $g_T^\Lambda = L_T$ by performing an exhaustive search over a specified polynomial-size range of possible values. If it succeeds, then it outputs Λ. Otherwise, it outputs $\perp$ indicating failure.

We emphasize that the polynomial running time of our decryption algorithm is guaranteed by restricting the output to lie within a fixed polynomial-size range. Note that similar exhaustive search step is used to determine the output in the decryption algorithm of all bilinear-map-based IPE constructions (both single and multi-input) available in the literature.

Remark 3.1: We would like to mention here that the FH-MIPE scheme described above can be proven to achieve the full-hiding security only when the adversary makes at least one ciphertext query for each of the n encryption indices, i.e., the restriction Eq. (2.1) is applicable. However, using a semantically secure SKE scheme, one can generically transform any FH-MIPE scheme that achieves full-hiding security under such restriction to one that achieves the full-hiding security even when the adversary makes no ciphertext query for some of the encryption slots. The transformation is rather straightforward and is presented in the full version of this paper.

■ **Correctness**: The correctness of the above FH-MIPE construction can be verified as follows: Observe that for any set of n ciphertexts $\{\text{CT}_\iota = (\iota, \boldsymbol{c}_\iota)\}_{\iota\in[n]}$, where $\text{CT}_\iota = (\iota, \boldsymbol{c}_\iota)$ encrypts some vector $\vec{x}_\iota \in \mathbb{F}_q^m$ with respect to the index ι, for $\iota \in [n]$, and any decryption key $\text{SK} = \{\boldsymbol{k}_\iota\}_{\iota\in[n]}$ corresponding to a set of n vectors $\{\vec{y}_\iota\}_{\iota\in[n]}$ such that $\vec{y}_\iota \in \mathbb{F}_q^m$ for all $\iota \in [n]$, we have

$$L_T = \prod_{\iota\in[n]} e(\boldsymbol{c}_\iota, \boldsymbol{k}_\iota) = g_T^{\sum\limits_{\iota\in[n]} \vec{x}_\iota \cdot \vec{y}_\iota}.$$

This follows from the expressions of $\boldsymbol{c}_\iota, \boldsymbol{k}_\iota$, for $\iota \in [n]$, in conjunction with the fact that for each $\iota \in [n]$, $\mathbb{B}_\iota$ and $\mathbb{B}_\iota^*$ are dual orthogonal bases. Thus, if $\sum\limits_{\iota\in[n]} \vec{x}_\iota \cdot \vec{y}_\iota$ is contained within the specified polynomial-size range of possible values that the decryption algorithm searches, then the decryption algorithm would definitely output $\Lambda = \sum\limits_{\iota\in[n]} \vec{x}_\iota \cdot \vec{y}_\iota$ as desired.

3.2 Security

Theorem 3.1 (Security of our FH-MIPE Scheme): *Assume that the k-LIN problem is hard. Then, the* FH-MIPE *construction described above achieves full-hiding security under the restriction that the adversary makes at least one ciphertext query for each encryption index. Additionally, assuming the existance of a semantically secure* SKE *scheme, we can generically convert the above* FH-MIPE *scheme to one that achieves full-hiding security without any restriction on the number of ciphertext queries per encryption slot. More formally, for any* PPT *adversary $\mathcal{A}$ against the full-hiding security of the* FH-MIPE *construction obtained by generically converting the above* FH-MIPE *scheme with the help of an* SKE *scheme, there exists a* PPT *algorithm $\mathcal{B}_1$ against the k-LIN problem and a* PPT *adversary $\mathcal{B}_2$ against the simantic security of* SKE *such that for any security parameter λ, we have*

$$\mathsf{Adv}_{\mathcal{A}}^{\text{FH-MIPE}}(\lambda) \leq [4 \sum_{\iota\in[n]} q_{\text{CT},\iota} + 2q_{\text{KEY}}]\mathsf{Adv}_{\mathcal{B}_1}^{k\text{-LIN}}(\lambda) + \mathsf{Adv}_{\mathcal{B}_2}^{\text{SKE}}(\lambda).$$

Proof: Here, we only proof the hull-hiding security of the above FH-MIPE scheme under the restriction that the adversary makes at least one ciphertext query per encryption slot. The proof is structured as a hybrid argument over a series of experiments which differ in the construction of the decryption keys and/or ciphertexts queried by the adversary $\mathcal{A}$ in the full-hiding security model

described in Definition 2.8. In the first hybrid experiment, the queried decryption keys and ciphertexts are constructed as those in the security experiment $\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(0)$. We then progressively change the ciphertexts and decryption keys in multiple hybrid steps to those in the security experiment $\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(1)$. We prove that each hybrid is indistinguishable from the previous one, thus proving the full-hiding security of the above FH-MIPE construction. Let q_{KEY} be the number of $\mathcal{A}$'s decryption key queries and $q_{\text{CT},\iota}$ (≥ 1), for $\iota \in [n]$, be the number of $\mathcal{A}$'s ciphertext queries for the ι^{th} index. As noted earlier, we consider $q_{\text{CT},\iota} \geq 1$ for all $\iota \in [n]$. The hybrid experiments are described below. In these hybrids, a part framed by a box indicates those terms which were modified in the transition from the previous game. The sequence of hybrid experiments follow:

► Sequence of Hybrid Experiments

Hyb_0: This experiment corresponds to the experiment $\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(0)$ described in Definition 2.8, i.e., the full-hiding security experiment where the random bit used by the challenger $\mathcal{B}$ to generate queried ciphertexts and decryption keys is $\beta = 0$. More precisely, for all $\iota \in [n], t_\iota \in [q_{\text{CT},\iota}]$, in response to the $t_\iota{}^{\text{th}}$ ciphertext query of $\mathcal{A}$ with respect to index ι corresponding to pair of vectors $(\vec{x}_{\iota,t_\iota,0}, \vec{x}_{\iota,t_\iota,1}) \in (\mathbb{F}_q^m)^2$, $\mathcal{B}$ returns $\text{CT}^*_{\iota,t_\iota} = (\iota, \boldsymbol{c}^*_{\iota,t_\iota})$, where

$$\boldsymbol{c}^*_{\iota,t_\iota} = (\vec{x}_{\iota,t_\iota,0}, \vec{0}^m, 1, \vec{0}^{k-1}, \varphi_{\iota,t_\iota,1}, \ldots, \varphi_{\iota,t_\iota,k}, 0)_{\mathbb{B}_\iota}, \tag{3.1}$$

and for all $i \in [q_{\text{KEY}}]$, to answer the i^{th} decryption key query of $\mathcal{A}$ corresponding to pair of sets of n vectors $(\{\vec{y}_{\iota,i,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,i,1}\}_{\iota\in[n]})$ such that $\vec{y}_{\iota,i,0}, \vec{y}_{\iota,i,1} \in \mathbb{F}_q^m$, $\mathcal{B}$ generates $\text{SK}^*_i = \{\boldsymbol{k}^*_{\iota,i}\}_{\iota\in[n]}$, where

$$\boldsymbol{k}^*_{\iota,i} = (\vec{y}_{\iota,i,0}, \vec{0}^m, r_{\iota,i}, \gamma_{\iota,i,1}, \ldots, \gamma_{\iota,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}^*_\iota}, \text{ for } \iota \in [n]. \tag{3.2}$$

Here, $\mathsf{params}_{\mathbb{G}} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\text{R}} \mathcal{G}_{\text{BPG}}()$; $\mathsf{params}_{\mathbb{V}} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\text{R}} \mathcal{G}_{\text{DPVS}}(2m + 2k + 1, \mathsf{params}_{\mathbb{G}})$; $\nu \xleftarrow{\text{U}} \mathbb{F}_q\backslash\{0\}$; $(\mathbb{B}_\iota, \mathbb{B}^*_\iota) \xleftarrow{\text{R}} \mathcal{G}_{\text{OB}}(2m + 2k+1, \mathsf{params}_{\mathbb{V}}, \nu)$, for $\iota \in [n]$; and $\varphi_{\iota,t_\iota,1}, \ldots, \varphi_{\iota,t_\iota,k}, r_{\iota,i}, \gamma_{\iota,i,1}, \ldots, \gamma_{\iota,i,k-1} \xleftarrow{\text{U}} \mathbb{F}_q$ for all $\iota \in [n], t_\iota \in [q_{\text{CT},\iota}], i \in [q_{\text{KEY}}]$, such that $\sum_{\iota\in[n]} r_{\iota,i} = 0$ for all $i \in [q_{\text{KEY}}]$.

Hyb_1 Sequence

$\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},1}$ ($\iota^* \in [n], \mu_{\iota^*} \in [q_{\text{ct},\iota^*}]$): $\mathsf{Hyb}_{1,0,q_{\text{CT},0},3}$ coincides with Hyb_0. This experiment is the same as $\mathsf{Hyb}_{1,\iota^*-1,q_{\text{CT},\iota^*-1},3}$, if $\mu_{\iota^*} = 1$, or $\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*}-1,3}$, if $\mu_{\iota^*} > 1$, with the only exception that in response to the $\mu_{\iota^*}{}^{\text{th}}$ ciphertext query of $\mathcal{A}$ with respect to index ι^* corresponding to pair of vectors $(\vec{x}_{\iota^*,\mu_{\iota^*},0}, \vec{x}_{\iota^*,\mu_{\iota^*},1}) \in (\mathbb{F}_q^m)^2$, $\mathcal{B}$ returns $\text{CT}^*_{\iota^*,\mu_{\iota^*}} = (\iota^*, \boldsymbol{c}^*_{\iota^*,\mu_{\iota^*}})$, where

$$\boldsymbol{c}^*_{\iota^*,\mu_{\iota^*}} = (\vec{x}_{\iota^*,\mu_{\iota^*},0}, \vec{0}^m, 1, \vec{0}^{k-1}, \varphi_{\iota^*,\mu_{\iota^*},1}, \ldots, \varphi_{\iota^*,\mu_{\iota^*},k}, \boxed{\rho_{\iota^*,\mu_{\iota^*}}})_{\mathbb{B}_{\iota^*}}. \tag{3.3}$$

Here, $\rho_{\iota^*,\mu_{\iota^*}} \xleftarrow{\text{U}} \mathbb{F}_q\backslash\{0\}$, and the other variables are formed as in $\mathsf{Hyb}_{1,\iota^*-1,q_{\text{CT},\iota^*-1},3}$ or $\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*}-1,3}$ according as $\mu_{\iota^*} = 1$ or $\mu_{\iota^*} > 1$.

$\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},2}$ ($\iota^* \in [n], \mu_{\iota^*} \in [q_{\text{ct},\iota^*}]$): This experiment is analogous to $\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},1}$ except that to answer the $\mu_{\iota^*}{}^{\text{th}}$ ciphertext query of $\mathcal{A}$ with respect to index ι^* corresponding to pair of vectors $(\vec{x}_{\iota^*,\mu_{\iota^*},0}, \vec{x}_{\iota^*,\mu_{\iota^*},1}) \in (\mathbb{F}_q^m)^2$,

$\mathcal{B}$ generates $\mathrm{CT}^*_{\iota^*,\mu_{\iota^*}} = (\iota^*, \boldsymbol{c}^*_{\iota^*,\mu_{\iota^*}})$, where

$$\boldsymbol{c}^*_{\iota^*,\mu_{\iota^*}} = (\vec{x}_{\iota^*,\mu_{\iota^*},0}, \boxed{\vec{x}_{\iota^*,\mu_{\iota^*},1}}, 1, \vec{0}^{k-1}, \varphi_{\iota^*,\mu_{\iota^*},1}, \ldots, \varphi_{\iota^*,\mu_{\iota^*},k}, \rho_{\iota^*,\mu_{\iota^*}})_{\mathbb{B}_{\iota^*}}. \quad (3.4)$$

Here, all the variables are created as in $\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},1}$.

$\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},3}$ ($\iota^* \in [n], \mu_{\iota^*} \in [q_{\mathsf{ct},\iota^*}]$): This experiment is exactly identical to $\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},2}$ with the only exception that in response to the $\mu_{\iota^*}{}^{\text{th}}$ ciphertext query of $\mathcal{A}$ with respect to the index ι^* corresponding to pair of vectors $(\vec{x}_{\iota^*,\mu_{\iota^*},0}, \vec{x}_{\iota^*,\mu_{\iota^*},1}) \in (\mathbb{F}_q^m)^2$, $\mathcal{B}$ returns $\mathrm{CT}^*_{\iota^*,\mu_{\iota^*}} = (\iota^*, \boldsymbol{c}^*_{\iota^*,\mu_{\iota^*}})$, where

$$\boldsymbol{c}^*_{\iota^*,\mu_{\iota^*}} = (\vec{x}_{\iota^*,\mu_{\iota^*},0}, \vec{x}_{\iota^*,\mu_{\iota^*},1}, 1, \vec{0}^{k-1}, \varphi_{\iota^*,\mu_{\iota^*},1}, \ldots, \varphi_{\iota^*,\mu_{\iota^*},k}, \boxed{0})_{\mathbb{B}_{\iota^*}}. \quad (3.5)$$

Here, all the variables are created as in $\mathsf{Hyb}_{1,\iota^*,\mu_{\iota^*},2}$.

Hyb_2 Sequence

$\mathsf{Hyb}_{2,\upsilon,1}$ ($\upsilon \in [q_{\mathsf{key}}]$): $\mathsf{Hyb}_{2,0,3}$ coincides with $\mathsf{Hyb}_{1,n,q_{\mathrm{CT},n},3}$. This experiment is analogous to $\mathsf{Hyb}_{2,\upsilon-1,3}$ with the only exception that in response to the υ^{th} decryption key query of $\mathcal{A}$ corresponding to the pair of sets of n vectors $(\{\vec{y}_{\iota,\upsilon,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,\upsilon,1}\}_{\iota\in[n]})$ such that $\vec{y}_{\iota,\upsilon,0}, \vec{y}_{\iota,\upsilon,1} \in \mathbb{F}_q^m$ for all $\iota \in [n]$, $\mathcal{B}$ gives back $\mathrm{SK}^*_\upsilon = \{\boldsymbol{k}^*_{\iota,\upsilon}\}_{\iota\in[n]}$, where

$$\boldsymbol{k}^*_{\iota,\upsilon} = (\vec{y}_{\iota,\upsilon,0}, \vec{0}^m, r_{\iota,\upsilon}, \gamma_{\iota,\upsilon,1}, \ldots, \gamma_{\iota,\upsilon,k-1}, \vec{0}^k, \boxed{\omega_{\iota,\upsilon}})_{\mathbb{B}^*_\iota}, \text{ for } \iota \in [n]. \quad (3.6)$$

Here, $\omega_{\iota,\upsilon} \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$ for all $\iota \in [n]$, such that $\sum_{\iota\in[n]} \omega_{\iota,\upsilon} = 0$, and all the other variables are generated as in $\mathsf{Hyb}_{2,\upsilon-1,3}$.

$\mathsf{Hyb}_{2,\upsilon,2}$ ($\upsilon \in [q_{\mathsf{key}}]$): This experiment is identical to $\mathsf{Hyb}_{2,\upsilon,1}$ except that in response to the υ^{th} decryption key query of $\mathcal{A}$ corresponding to the pair of sets of n vectors $(\{\vec{y}_{\iota,\upsilon,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,\upsilon,1}\}_{\iota\in[n]})$ such that $\vec{y}_{\iota,\upsilon,0}, \vec{y}_{\iota,\upsilon,1} \in \mathbb{F}_q^m$, $\mathcal{B}$ returns $\mathrm{SK}^*_\upsilon = \{\boldsymbol{k}^*_{\iota,\upsilon}\}_{\iota\in[n]}$, where

$$\boldsymbol{k}^*_{\iota,\upsilon} = (\boxed{\vec{0}^m, \vec{y}_{\iota,\upsilon,1}, \widetilde{r}_{\iota,\upsilon}}, \gamma_{\iota,\upsilon,1}, \ldots, \gamma_{\iota,\upsilon,k-1}, \vec{0}^k, \omega_{\iota,\upsilon})_{\mathbb{B}^*_\iota}, \text{ for } \iota \in [n]. \quad (3.7)$$

Here, $\widetilde{r}_{\iota,\upsilon} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for all $\iota \in [n]$, such that $\sum_{\iota\in[n]} \widetilde{r}_{\iota,\upsilon} = 0$, and all the variables are generated as in $\mathsf{Hyb}_{2,\upsilon,1}$.

$\mathsf{Hyb}_{2,\upsilon,3}$ ($\upsilon \in [q_{\mathsf{key}}]$): This experiment is analogous to $\mathsf{Hyb}_{2,\upsilon,2}$ except that to answer the υ^{th} decryption key query of $\mathcal{A}$ corresponding to the pair of sets of n vectors $(\{\vec{y}_{\iota,\upsilon,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,\upsilon,1}\}_{\iota\in[n]})$ such that $\vec{y}_{\iota,\upsilon,0}, \vec{y}_{\iota,\upsilon,1} \in \mathbb{F}_q^m$, $\mathcal{B}$ gives back $\mathrm{SK}^*_\upsilon = \{\boldsymbol{k}^*_{\iota,\upsilon}\}_{\iota\in[n]}$, where

$$\boldsymbol{k}^*_{\iota,\upsilon} = (\vec{0}^m, \vec{y}_{\iota,\upsilon,1}, \widetilde{r}_{\iota,\upsilon}, \gamma_{\iota,\upsilon,1}, \ldots, \gamma_{\iota,\upsilon,k-1}, \vec{0}^k, \boxed{0})_{\mathbb{B}^*_\iota}, \text{ for } \iota \in [n]. \quad (3.8)$$

Here, all the variables are generated as in $\mathsf{Hyb}_{2,\upsilon,2}$.

Hyb_3: This experiment is identical to $\mathsf{Hyb}_{2,q_{\mathrm{KEY}},3}$ with the only exception that for all $\iota \in [n], t_\iota \in [q_{\mathrm{CT},\iota}]$, in response to the $t_\iota{}^{\text{th}}$ ciphertext query of $\mathcal{A}$ with respect to index ι corresponding to pair of vectors $(\vec{x}_{\iota,t_\iota,0}, \vec{x}_{\iota,t_\iota,1}) \in (\mathbb{F}_q^m)^2$, $\mathcal{B}$ returns $\mathrm{CT}^*_{\iota,t_\iota} = (\iota, \boldsymbol{c}^*_{\iota,t_\iota})$, where

$$\boldsymbol{c}^*_{\iota,t_\iota} = (\boxed{\vec{x}_{\iota,t_\iota,1}, \vec{x}_{\iota,t_\iota,0}}, 1, \vec{0}^{k-1}, \varphi_{\iota,t_\iota,1}, \ldots, \varphi_{\iota,t_\iota,k}, 0)_{\mathbb{B}_\iota}, \quad (3.9)$$

and for all $i \in [q_{\mathrm{KEY}}]$, to answer the i^{th} decryption key query of $\mathcal{A}$ corresponding to pair of sets of n vectors $(\{\vec{y}_{\iota,i,0}\}_{\iota\in[n]}, \{\vec{y}_{\iota,i,1}\}_{\iota\in[n]})$ such that $\vec{y}_{\iota,i,0}, \vec{y}_{\iota,i,1} \in \mathbb{F}_q^m$,

$\mathcal{B}$ generates $\mathrm{SK}_i^* = \{\boldsymbol{k}_{\iota,i}^*\}_{\iota\in[n]}$, where

$$\boldsymbol{k}_{\iota,i}^* = (\boxed{\vec{y}_{\iota,i,1}, \vec{0}^m}, \widetilde{r}_{\iota,i}, \gamma_{\iota,i,1}, \ldots, \gamma_{\iota,i,k-1}, \vec{0}^k, 0)_{\mathbb{B}_\iota^*}, \text{ for } \iota \in [n]. \tag{3.10}$$

Here, all the variables are generated as in $\mathsf{Hyb}_{2,q_{\mathrm{KEY}},3}$.

Hyb_4: This experiment corresponds to the experiment $\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(1)$ described in Definition 2.8, i.e., the full-hiding security experiment where the random bit used by $\mathcal{B}$ to generate the ciphertexts and decryption keys queried by $\mathcal{A}$ is $\beta = 1$.

► **Analysis**

Let us now denote by $\mathsf{Adv}_{\mathcal{A}}^{(h)}(\lambda)$ the advantage of the adversary $\mathcal{A}$, i.e., $\mathcal{A}$'s probability of outputting 1 in Hyb_h, for $h \in \{0, \{1, \iota^*, \mu_{\iota^*}, \jmath\}_{\iota^*\in[n],\mu_{\iota^*}\in[q_{\mathrm{CT},\iota^*}],\jmath\in[3]}, \{2, \upsilon, \jmath\}_{\upsilon\in[q_{\mathrm{KEY}}],\jmath\in[3]}, 3, 4\}$. Then, by the definitions of hybrids, we clearly have $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) \equiv \mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(0) = 1]$, $\mathsf{Adv}_{\mathcal{A}}^{(1,0,q_{\mathrm{CT},0},3)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(2,0,3)}(\lambda) \equiv \mathsf{Adv}_{\mathcal{A}}^{(1,n,q_{\mathrm{CT},n},3)}(\lambda)$, and $\mathsf{Adv}_{\mathcal{A}}^{(4)}(\lambda) \equiv \mathsf{Pr}[\mathsf{Expt}_{\mathcal{A}}^{\text{FH-MIPE}}(1) = 1]$. Also, observe that the transition from Hyb_3 to Hyb_4 is essentially the reverse transition of the Hyb_1 sequence of hybrids with $\vec{x}_{\iota^*,\mu_{\iota^*},0}$ and $\vec{x}_{\iota^*,\mu_{\iota^*},1}$ interchanged. Therefore, it follows that

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\text{FH-MIPE}}(\lambda) \leq 2 \sum_{\iota^*\in[n]} \big[& |\mathsf{Adv}_{\mathcal{A}}^{(1,\iota^*-1,q_{\mathrm{CT},\iota^*-1},3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1,\iota^*,1,1)}(\lambda)| \\
& + \sum_{\mu_{\iota^*}\in[2,q_{\mathrm{CT},\iota^*}]} |\mathsf{Adv}_{\mathcal{A}}^{(1,\iota^*,\mu_{\iota^*}-1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1,\iota^*,\mu_{\iota^*},1)}(\lambda)| \\
& + \sum_{\mu_{\iota^*}\in[q_{\mathrm{CT},\iota^*}],\jmath\in[2,3]} |\mathsf{Adv}_{\mathcal{A}}^{(1,\iota^*,\mu_{\iota^*},\jmath-1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1,\iota^*,\mu_{\iota^*},\jmath)}(\lambda)| \big] \\
& + \sum_{\upsilon\in[q_{\mathrm{KEY}}]} \big[|\mathsf{Adv}_{\mathcal{A}}^{(2,\upsilon-1,3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2,\upsilon,1)}(\lambda)| \\
& + \sum_{\jmath\in[2,3]} |\mathsf{Adv}_{\mathcal{A}}^{(2,\upsilon,\jmath-1)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2,\upsilon,\jmath)}(\lambda)| \big] \\
& + |\mathsf{Adv}_{\mathcal{A}}^{(2,q_{\mathrm{KEY}},3)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)|.
\end{aligned} \tag{3.11}$$

The fact that each term on the RHS of Eq. (3.11) is negligible is formally argued in a sequence of lemmas presented in the full version of this paper. This completes the proof of Theorem 3.1. □

4 The Proposed Full-Hiding Unbounded Multi-Input Inner Product Functional Encryption Scheme

In this section, we present our FH-UMIPE scheme.

4.1 Construction

For the simplicity, we consider the scheme based on the SXDH(1-Lin) in this section. However, it is clear that we can instantiate our FH-UMIPE scheme from k-Lin assumption. We also consider the case where the vector length m is poly-

nomial in λ. Let $F_1 : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \mathbb{F}_q^{(2m+3)\times(2m+3)}$ and $F_2 : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be pseudorandom functions and (SKE.KeyGen, SKE.Encrypt, SKE.Decrypt) be a semantically secure secret key encryption scheme whose secret key space is $\{0,1\}^\lambda$. We require that SKE.KeyGen outputs a randomly chosen λ-bit string as a secret key K, i.e., $K \xleftarrow{\mathsf{U}} \{0,1\}^\lambda$. We abuse the notation such that for a set of N vectors of M dimensional DPVS $\mathbb{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_N)$ and $W \in \mathsf{GL}(M, \mathbb{F}_q)$, $\mathbb{B} = \mathbb{D}W$ denotes $\mathbb{B} = (\boldsymbol{d}_1 W, \ldots, \boldsymbol{d}_N W)$.

FH-UMIPE.Setup$(m, \mathscr{B})$: It takes as input the unary encoded security parameter 1^λ, the length $m \in \mathbb{N}$ of vectors, and the bound $\mathscr{B} \in \mathbb{N}$. It proceeds as follows:

1. First, it generates a bilinear group $\mathsf{params}_\mathbb{G} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{BPG}}()$ with q a λ-bit prime.
2. Next, it forms $\mathsf{params}_\mathbb{V} = (q, \mathbb{V}_1, \mathbb{V}_2, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_2, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{DPVS}}(2m+3, \mathsf{params}_\mathbb{G})$, samples $\nu \xleftarrow{\mathsf{U}} \mathbb{F}_q \backslash \{0\}$, computes $g_T = e(g_1, g_2)^\nu$, generates $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathrm{OB}}(2m+3, \mathsf{params}_\mathbb{V}, \nu)$, and samples PRF keys $K_1, K_2 \xleftarrow{\mathsf{U}} \{0,1\}^\lambda$. Then it sets $\widehat{\mathbb{D}} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_m, \boldsymbol{d}_{2m+1}, \boldsymbol{d}_{2m+2}), \widehat{\mathbb{D}}^* = (\boldsymbol{d}^*_1, \ldots, \boldsymbol{d}^*_m, \boldsymbol{d}^*_{2m+1})$.
3. It publishes the public parameters $\mathrm{PP} = (\mathsf{params}_\mathbb{V}, g_T)$, while keeps the master secret key $\mathrm{MSK} = (K_1, K_2, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$.

FH-UMIPE.KeyGen$(\mathrm{PP}, \mathrm{MSK}, S, \{\vec{y}_\iota\}_{\iota \in S})$: On input the public parameters PP, the master secret key MSK, a set of indices $S \subseteq [t(\lambda)]$ for any polynomial t, along with a $|S|$-tuple of vectors $\{\vec{y}_\iota\}_{\iota \in S} \in (\mathbb{Z}^m)^{|S|}$, this algorithm executes the following steps:

1. First, it creates random dual orthogonal bases for the index $\iota \in S$ as follows;
$$W_\iota = F_1(K_1, \iota), \ \ \mathbb{B}^*_\iota = \mathbb{D}^* W^*_\iota.$$
If W_ι for some $\iota \in S$ is not a regular matrix, then it outputs $\perp$ and halts.
2. Next, for each $\iota \in S$, it computes decryption keys similarly to the bounded case;
$$\{r_\iota\}_{\iota \in S} \xleftarrow{\mathsf{U}} \mathbb{F}_q \ \text{ s.t. } \sum_{\iota \in S} r_\iota = 0, \quad \boldsymbol{k}_\iota = (\vec{y}_\iota, \vec{0}^m, r_\iota, \vec{0}^2)_{\mathbb{B}^*_\iota}.$$
3. Let s_j be the j^{th} element of S in ascending order. Then it iteratively encrypts the decryption keys by symmetric key encryption as
$$\begin{aligned} C_1 &= \mathsf{SKE.Encrypt}(F_2(K_2, s_1), \{\boldsymbol{k}_\iota\}_{\iota \in S}), \\ C_2 &= \mathsf{SKE.Encrypt}(F_2(K_2, s_2), C_1), \\ &\vdots \\ C_{|S|} &= \mathsf{SKE.Encrypt}(F_2(K_2, s_{|S|}), C_{|S|-1}), \end{aligned}$$
and outputs $\mathrm{SK}_S = (C_{|S|}, S)$ as a decryption key for FH-UMIPE.

FH-UMIPE.Encrypt$(\mathrm{PP}, \mathrm{MSK}, \iota, \vec{x}_\iota)$: Taking as input the public parameters PP, the master secret key MSK, an index $\iota \in [2^\lambda]$, along with a vector $\vec{x}_\iota \in \mathbb{Z}^m$, this algorithm performs the following steps:

1. First, it creates random dual orthogonal bases for the index ι as follows;
$$W_\iota = F_1(K_1, \iota), \ \ \mathbb{B}_\iota = \mathbb{D} W_\iota.$$

If W_ι is not a regular matrix, then it outputs $\perp$ and halts.

2. Otherwise, it selects random $\kappa_\iota \xleftarrow{\mathsf{U}} \mathbb{F}_q$, computes
$$\boldsymbol{c}_\iota = (\vec{x}_\iota, \vec{0}^m, 1, \kappa_\iota, 0)_{\mathbb{B}_\iota}, \quad k_\iota = F_2(K_2, \iota),$$
and outputs the ciphertext $\mathrm{CT}_\iota = (\boldsymbol{c}_\iota, k_\iota, \iota)$.

$\mathsf{FH\text{-}UMIPE.Decrypt}(\mathrm{PP}, \mathrm{SK}_S, \{\mathrm{CT}_\iota\}_{\iota \in S})$: A decrypter takes as input the public parameters PP, a decryption key SK_S for a set S, and a tuple of $|S|$ ciphertexts $\{\mathrm{CT}_\iota\}_{\iota \in S}$. It does the following:

1. It first decrypts decryption keys as follows;
$$\begin{aligned} C'_{|S|-1} &= \mathsf{SKE.Decrypt}(k_{s_{|S|}}, C_{|S|}), \\ &\vdots \\ C'_1 &= \mathsf{SKE.Decrypt}(k_{s_2}, C'_2), \\ \{\boldsymbol{k}'_\iota\}_{\iota \in S} &= \mathsf{SKE.Decrypt}(k_{s_1}, C'_1). \end{aligned}$$
2. Next, it computes $L_T = \prod_{\iota \in S} e(\boldsymbol{c}_\iota, \boldsymbol{k}'_\iota)$.
3. Then, it attempts to determine a value $\Lambda \in \mathbb{N}$ such that $g_T^\Lambda = L_T$ by performing an exhaustive search over a specified polynomial-size range of possible values. If it succeeds, then it outputs Λ. Otherwise, it outputs $\perp$ indicating failure.

■ **Correctness**: The correctness of our unbounded scheme is presented in the full version of this paper.

4.2 Security

Theorem 4.1 (Security of Our FH-UMIPE Scheme): *Assume that F_1 and F_2 are pseudorandom functions,* SKE *is semantically secure symmetric key encryption, and* SXDH *problem is hard, then our* FH-UMIPE *construction achieves full-hiding security. More formally, for any* PPT *adversary $\mathcal{A}$ against the full-hiding security of the proposed* FH-UMIPE *construction, there exists a* PPT *algorithm $\mathcal{B}_1$ against the* SXDH *problem, $\mathcal{B}_2$ against the symmetric key encryption scheme, and $\mathcal{B}_3$ and $\mathcal{B}_4$ against the pseudorandom functions such that for any security parameter λ, we have*

$$\begin{aligned} \mathsf{Adv}^{\text{FH-UMIPE}}_{\mathcal{A}}(\lambda) \leq [4 \sum_{\iota \in [2^\lambda]} q_{\mathrm{CT},\iota} + 2q_{\mathrm{KEY}}]\mathsf{Adv}^{\text{SXDH}}_{\mathcal{B}_1}(\lambda) + n_{max} q_{\mathrm{KEY}} \mathsf{Adv}^{\text{SKE}}_{\mathcal{B}_2}(\lambda) \\ + 2\mathsf{Adv}^{\text{PRF}_1}_{\mathcal{B}_3}(\lambda) + 2\mathsf{Adv}^{\text{PRF}_2}_{\mathcal{B}_4}(\lambda), \end{aligned}$$

where $q_{\mathrm{CT},\iota}$ is the total number of ciphertext query for the index ι, q_{KEY} is the total number of decryption key query, and n_{max} is the maximum index of a decryption key that $\mathcal{A}$ queries, i.e., $S_i \subseteq [n_{max}]$ for all $i \in [q_{\mathrm{SK}}]$.

Proof: The proof of Theorem 4.1 is structured as a hybrid argument over a series of experiments which differ in the construction of the decryption keys and/or ciphertexts queried by the adversary $\mathcal{A}$ in the full-hiding security model described in Definition 2.9. The hybrid transition is proceeded in the similar way to the bounded scheme, that is, first we gradually change the ciphertext form

from $(\vec{x}_{\iota,0}, \vec{0}^m, 1, \kappa_\iota, 0)_{\mathbb{B}_\iota}$ to $(\vec{x}_{\iota,0}, \vec{x}_{\iota,1}, 1, \kappa_\iota, 0)_{\mathbb{B}_\iota}$. Next, we change the decryption key form from $(\vec{y}_{\iota,0}, \vec{0}^m, r_\iota, \vec{0}^2)_{\mathbb{B}^*_\iota}$ to $(\vec{0}^m, \vec{y}_{\iota,1}, r_\iota, \vec{0}^2)_{\mathbb{B}^*_\iota}$. Then, switch the first m coefficients with the second m coefficients and restore the ciphertexts. The proof of the ciphertexts part is almost same as that of the bounded scheme, while the decryption key part is more complicated than the bounded one. The hybrid experiments are described below. In these hybrids, a part framed by a box indicates those terms which were modified in the transition from the previous game. The sequence of hybrid experiments follow:

► **Sequence of Hybrid Experiments**

$\mathsf{Hyb}_\mathbf{0}$: We denote the j^{th} element of S_i in ascending order by $s_{i,j}$. This experiment is the same as $\mathsf{Expt}^{\text{FH-UMIPE}}_{\mathcal{A}}(0)$ defined in Definition 2.9. That is, when the challenger receives $(\vec{x}_{\iota,t_\iota,0}, \vec{x}_{\iota,t_\iota,1})$ from $\mathcal{A}$ as a $t_\iota{}^{\text{th}}$ ciphertext query for index ι, it returns $\text{CT}^*_{\iota,t_\iota} = (\boldsymbol{c}^*_{\iota,t_\iota}, k_\iota, \iota)$, where

$$W_\iota = F_1(K_1, \iota), \;\; \mathbb{B}_\iota = \mathbb{D}W_\iota,$$

$$\kappa_{\iota,t_\iota} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \quad \boldsymbol{c}^*_{\iota,t_\iota} = (\vec{x}_{\iota,t_\iota,0}, \vec{0}^m, 1, \kappa_{\iota,t_\iota}, 0)_{\mathbb{B}_\iota}, \quad k_\iota = F_2(K_2, \iota).$$

On the other hand, when the challenger receives $(S_i, \{\vec{y}_{\iota,i,0}, \vec{y}_{\iota,i,1}\}_{\iota\in S_i})$ for i^{th} decryption key query, it returns $\text{SK}^*_{S_i,i} = (C_{|S_i|}, S_i)$, where

$$r_{\iota,i} \xleftarrow{\mathsf{U}} \mathbb{F}_q \;\text{ s.t. }\; \sum_{\iota\in S_i} r_{\iota,i} = 0, \;\; W_\iota = F_1(K_1, \iota), \;\; \mathbb{B}^*_\iota = \mathbb{D}^* W^*_\iota,$$

$$\boldsymbol{k}^*_{\iota,i} = (\vec{y}_{\iota,i,0}, \vec{0}^m, r_{\iota,i}, \vec{0}^2)_{\mathbb{B}^*_\iota} \quad \text{for } \iota \in S_i,$$

$$C_{|S_i|} = \mathsf{SKE.Encrypt}(F_2(K_2, s_{i,|S_i|}), \ldots, \mathsf{SKE.Encrypt}(F_2(K_2, s_{i,1}), \{\boldsymbol{k}^*_{\iota,i}\}_{\iota\in S_i}) \ldots).$$

$\mathsf{Hyb}_\mathbf{1}$: In this hybrids, we replace pseudorandom functions $F_i(K_i, \cdot)$ for $i \in \{1, 2\}$ with random functions $R_i(\cdot) \xleftarrow{\mathsf{U}} \mathcal{R}_{i,\lambda}$, where $\mathcal{R}_{i,\lambda}$ is a set of functions consists of all functions that have the same domain and range as F_i. Observe that all dual orthogonal bases used in the ciphertexts and decryption keys queried by $\mathcal{A}$ are completely independent and random ones by each index after Hyb_1.

$\mathsf{Hyb}_\mathbf{2}$: The all replies for the ciphertext queries are changed as follows;

$$W_\iota = R_1(\iota), \;\; \mathbb{B}_\iota = \mathbb{D}W_\iota,$$

$$\kappa_{\iota,t_\iota} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \quad \boldsymbol{c}^*_{\iota,t_\iota} = (\vec{x}_{\iota,t_\iota,0}, \boxed{\vec{x}_{\iota,t_\iota,1}}, 1, \kappa_{\iota,t_\iota}, 0)_{\mathbb{B}_\iota}, \quad k_\iota = R_2(\iota),$$

and returns $\text{CT}^*_{\iota,t_\iota} = (\boldsymbol{c}^*_{\iota,t_\iota}, k_\iota, \iota)$.

$\mathsf{Hyb}_\mathbf{3}$ Sequence

$\mathsf{Hyb}_{\mathbf{3},\upsilon}$ ($\upsilon \in [q_{\mathsf{key}}]$): $\mathsf{Hyb}_{3,0}$ is the same as Hyb_2. The challenger replies to the first υ decryption key queries, i.e., the i^{th} decryption key query for all $i \le \upsilon$, as

$$r_{\iota,i} \xleftarrow{\mathsf{U}} \mathbb{F}_q \;\text{ s.t. }\; \sum_{\iota\in S_i} r_{\iota,i} = 0, \;\; W_\iota = R_1(\iota), \;\; \mathbb{B}^*_\iota = \mathbb{D}^* W^*_\iota,$$

$$\boldsymbol{k}^*_{\iota,i} = (\vec{0}^m, \boxed{\vec{y}_{\iota,i,1}}, r_{\iota,i}, \vec{0}^2)_{\mathbb{B}^*_\iota} \quad \text{for } \iota \in S_i,$$

$$C_{|S_i|} = \mathsf{SKE.Encrypt}(R_2(s_{i,|S_i|}), \ldots, \mathsf{SKE.Encrypt}(R_2(s_{i,1}), \{\boldsymbol{k}^*_{\iota,i}\}_{\iota\in S_i}) \ldots),$$

and returns $\text{SK}^*_{S_i,i} = (C_{|S_i|}, S_i)$. For the other decryption key queries, the challenger replies the same way as Hyb_2.

Hyb_4: In this hybrid, we switch the coefficients of 1 to m^{th} vector with those of $m+1$ to $2m^{\text{th}}$ vector in both decryption key side and ciphertext side. Namely, the replies for the ciphertext queries are $\mathrm{CT}^*_{\iota,t_\iota} = (\boldsymbol{c}^*_{\iota,t_\iota}, k_\iota, \iota)$, where

$$W_\iota = R_1(\iota), \;\; \mathbb{B}_\iota = \mathbb{D}W_\iota,$$

$$\kappa_{\iota,t_\iota} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \quad \boldsymbol{c}^*_{\iota,t_\iota} = (\boxed{\vec{x}_{\iota,t_\iota,1}, \vec{x}_{\iota,t_\iota,0}}, 1, \kappa_{\iota,t_\iota}, 0)_{\mathbb{B}_\iota}, \quad k_\iota = R_2(\iota),$$

and the replies for the decryption key queries are $\mathrm{SK}^*_{S_i,i} = (C_{|S_i|}, S_i)$, where

$$r_{\iota,i} \xleftarrow{\mathsf{U}} \mathbb{F}_q \;\text{ s.t. }\; \sum_{\iota \in S_i} r_{\iota,i} = 0, \;\; W_\iota = R_1(\iota), \;\; \mathbb{B}^*_\iota = \mathbb{D}^* W^*_\iota,$$

$$\boldsymbol{k}^*_{\iota,i} = (\boxed{\vec{y}_{\iota,i,1}, \vec{0}^m}, r_{\iota,i}, \vec{0}^2)_{\mathbb{B}^*_\iota} \quad \text{for } \iota \in S_i,$$

$$C_{|S_i|} = \mathsf{SKE.Encrypt}(R_2(s_{i,|S_i|}), \ldots, \mathsf{SKE.Encrypt}(R_2(s_{i,1}), \{\boldsymbol{k}^*_{\iota,i}\}_{\iota \in S_i}) \ldots).$$

Hyb_5: This hybrid is the same as $\mathsf{Expt}^{\text{FH-UMIPE}}_{\mathcal{A}}(1)$ defined in Definition 2.9. That is, the replies for the ciphertext queries are $\mathrm{CT}^*_{\iota,t_\iota} = (\boldsymbol{c}^*_{\iota,t_\iota}, k_\iota, \iota)$, where

$$W_\iota = \boxed{F_1(K_1, \iota)}, \;\; \mathbb{B}_\iota = \mathbb{D}W_\iota,$$

$$\kappa_{\iota,t_\iota} \xleftarrow{\mathsf{U}} \mathbb{F}_q, \quad \boldsymbol{c}^*_{\iota,t_\iota} = (\vec{x}_{\iota,t_\iota,1}, \boxed{\vec{0}^m}, 1, \kappa_{\iota,t_\iota}, 0)_{\mathbb{B}_\iota}, \quad \boxed{k_\iota = F_2(K_2, \iota)},$$

and the replies for the decryption key queries are $\mathrm{SK}^*_{S_i,i} = (C_{|S_i|}, S_i)$, where

$$r_{\iota,i} \xleftarrow{\mathsf{U}} \mathbb{F}_q \;\text{ s.t. }\; \sum_{\iota \in S_i} r_{\iota,i} = 0, \;\; \boxed{W_\iota = F_1(K_1, \iota)}, \;\; \mathbb{B}^*_\iota = \mathbb{D}^* W^*_\iota,$$

$$\boldsymbol{k}^*_{\iota,i} = (\vec{y}_{\iota,i,1}, \vec{0}^m, r_{\iota,i}, \vec{0}^2)_{\mathbb{B}^*_\iota} \quad \text{for } \iota \in S_i,$$

$$C_{|S_i|} = \mathsf{SKE.Encrypt}(\boxed{F_2(K_2, s_{i,|S_i|})}, \ldots, \mathsf{SKE.Encrypt}(\boxed{F_2(K_2, s_{i,1})}, \{\boldsymbol{k}^*_{\iota,i}\}_{\iota \in S_i}) \ldots).$$

► **Analysis**

Let us now denote by $\mathsf{Adv}^{(h)}_{\mathcal{A}}(\lambda)$ the advantage of the adversary $\mathcal{A}$, i.e., $\mathcal{A}$'s probability of outputting 1 in Hyb_h. Then, we can see that

$$\begin{aligned}\mathsf{Adv}^{\text{FH-UMIPE}}_{\mathcal{A}}(\lambda) \leq\; & |\mathsf{Adv}^{(0)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(1)}_{\mathcal{A}}(\lambda)| + |\mathsf{Adv}^{(1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(2)}_{\mathcal{A}}(\lambda)| \\ & + \sum_{\upsilon=1}^{q_{\text{KEY}}} |\mathsf{Adv}^{(3,\upsilon-1)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(3,\upsilon)}_{\mathcal{A}}(\lambda)| \qquad (4.1) \\ & + |\mathsf{Adv}^{(3,q_{\text{KEY}})}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(4)}_{\mathcal{A}}(\lambda)| + |\mathsf{Adv}^{(4)}_{\mathcal{A}}(\lambda) - \mathsf{Adv}^{(5)}_{\mathcal{A}}(\lambda)|.\end{aligned}$$

The fact that each term on the RHS of Eq. (4.1) is negligible is formally argued in a sequence of lemmas in the full version of this paper. This completes the proof of Theorem 4.1. □

References

1. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Public Key Cryptography–PKC 2015. pp. 733–751. Springer (2015)
2. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: Function-hiding realizations and constructions without pairings. Cryptology ePrint Archive, Report 2017/972 (2017)

3. Abdalla, M., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Advances in Cryptology–EUROCRYPT 2017. pp. 601–626. Springer (2017)
4. Agrawal, S., Agrawal, S., Badrinarayanan, S., Kumarasubramanian, A., Prabhakaran, M., Sahai, A.: Function private functional encryption and property preserving encryption: New definitions and positive results. Cryptology ePrint Archive, Report 2013/744 (2013)
5. Ananth, P., Brakerski, Z., Segev, G., Vaikuntanathan, V.: From selective to adaptive security in functional encryption. In: Advances in Cryptology–CRYPTO 2015. pp. 657–677. Springer (2015)
6. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Advances in Cryptology–CRYPTO 2015. pp. 308–326. Springer (2015)
7. Badrinarayanan, S., Gupta, D., Jain, A., Sahai, A.: Multi-input functional encryption for unbounded arity functions. In: Advances in Cryptology–ASIACRYPT 2015. pp. 27–51. Springer (2015)
8. Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S., Yang, K.: On the (im) possibility of obfuscating programs. In: Advances in Cryptology–CRYPTO 2001. pp. 1–18. Springer (2001)
9. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Advances in Cryptology–ASIACRYPT 2015. pp. 470–491. Springer (2015)
10. Boneh, D., Raghunathan, A., Segev, G.: Function-private identity-based encryption: Hiding the function in functional encryption. In: Advances in Cryptology–CRYPTO 2013, pp. 461–478. Springer (2013)
11. Boneh, D., Raghunathan, A., Segev, G.: Function-private subspace-membership encryption and its applications. In: Advances in Cryptology–ASIACRYPT 2013. pp. 255–275. Springer (2013)
12. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. Theory of Cryptography–TCC 2011 pp. 253–273 (2011)
13. Brakerski, Z., Komargodski, I., Segev, G.: Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In: Advances in Cryptology–EUROCRYPT 2016. pp. 852–880. Springer (2016)
14. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Theory of Cryptography Conference–TCC 2015. pp. 306–324. Springer (2015)
15. Cash, D., Liu, F.H., O'Neill, A., Zhang, C.: Reducing the leakage in practical order-revealing encryption. Cryptology ePrint Archive, Report 2016/661 (2016)
16. Chenette, N., Lewi, K., Weis, S.A., Wu, D.J.: Practical order-revealing encryption with limited leakage. In: Fast Software Encryption–FSE 2016. pp. 474–493. Springer (2016)
17. Coron, J.S., Lepoint, T., Tibouchi, M.: Practical multilinear maps over the integers. In: Advances in Cryptology–CRYPTO 2013. pp. 476–493. Springer (2013)
18. Datta, P., Dutta, R., Mukhopadhyay, S.: Functional encryption for inner product with full function privacy. In: Public-Key Cryptography–PKC 2016. pp. 164–195. Springer (2016)
19. Garg, S., Gentry, C., Halevi, S.: Candidate multilinear maps from ideal lattices. In: Advances in Cryptology–EUROCRYPT 2013. pp. 1–17. Springer (2013)
20. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: SIAM Journal on Computing. vol. 45, pp. 882–929. SIAM (2016)

21. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Theory of Cryptography Conference–TCC 2016. pp. 480–511. Springer (2016)
22. Garg, S., Gentry, C., Sahai, A., Waters, B.: Witness encryption and its applications. In: symposium on Theory of computing–stoc 2013. pp. 467–476. ACM (2013)
23. Goldwasser, S., Gordon, S.D., Goyal, V., Jain, A., Katz, J., Liu, F.H., Sahai, A., Shi, E., Zhou, H.S.: Multi-input functional encryption. In: Advances in Cryptology–EUROCRYPT 2014. pp. 578–602. Springer (2014)
24. Goyal, V., Jain, A., O'Neill, A.: Multi-input functional encryption with unbounded-message security. In: Advances in Cryptology–ASIACRYPT 2016. pp. 531–556. Springer (2016)
25. Iovino, V., Tang, Q., Zebrowski, K.: On the power of public-key functional encryption with function privacy. Cryptology ePrint Archive, Report 2015/470 (2015)
26. Ishai, Y., Pandey, O., Sahai, A.: Public-coin differing-inputs obfuscation and its applications. In: Theory of Cryptography Conference–TCC 2015. pp. 668–697. Springer (2015)
27. Kim, S., Lewi, K., Mandal, A., Montgomery, H.W., Roy, A., Wu, D.J.: Function-hiding inner product encryption is practical. Cryptology ePrint Archive, Report 2016/440 (2016)
28. Komargodski, I., Segev, G.: From minicrypt to obfustopia via private-key functional encryption. In: Advances in Cryptology–EUROCRYPT 2017. pp. 122–151. Springer (2017)
29. Komargodski, I., Segev, G., Yogev, E.: Functional encryption for randomized functionalities in the private-key setting from minimal assumptions. In: Theory of Cryptography Conference–TCC 2015. pp. 352–377. Springer (2015)
30. Lee, K., Lee, D.H.: Two-input functional encryption for inner products from bilinear maps. Cryptology ePrint Archive, Report 2016/432 (2016)
31. Lewi, K., Wu, D.J.: Order-revealing encryption: New constructions, applications, and lower bounds. In: ACM SIGSAC Conference on Computer and Communications Security–CCS 2016. pp. 1167–1178. ACM (2016)
32. Lin, H.: Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. In: Advances in Cryptology–CRYPTO 2017. pp. 599–629. Springer (2017)
33. Lin, H., Vaikuntanathan, V.: Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In: Foundations of Computer Science–FOCS 2016. pp. 11–20. IEEE (2016)
34. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Advances in Cryptology–ASIACRYPT 2009. pp. 214–231. Springer (2009)
35. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Advances in Cryptology–CRYPTO 2010. pp. 191–208. Springer (2010)
36. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010)
37. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. Cryptology ePrint Archive, Report 2007/074 (2007)
38. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Theory of Cryptography Conference–TCC 2009. pp. 457–473. Springer (2009)
39. Tomida, J., Abe, M., Okamoto, T.: Efficient functional encryption for inner-product values with full-hiding security. In: Information Security–ISC 2016. pp. 408–425. Springer (2016)