Multi-Input Quadratic Functional Encryption: Stronger Security, Broader Functionality

Shweta Agrawal[1], Rishab Goyal[2], and Junichi Tomida[3](✉)

[1] IIT Madras, Chennai, India
shweta.a@cse.iitm.ac.in
[2] UW-Madison, Madison, USA
rishab@cs.wisc.edu
[3] NTT Social Informatics Laboratories,Tokyo, Japan
tomida.junichi@gmail.com

Abstract. Multi-input functional encryption, MIFE, is a powerful generalization of functional encryption that allows computation on encrypted data coming from multiple different data sources. In a recent work, Agrawal, Goyal, and Tomida (CRYPTO 2021) constructed MIFE for the class of quadratic functions. This was the first MIFE construction from bilinear maps that went beyond inner product computation. We advance the state-of-the-art in MIFE, and propose new constructions with *stronger security* and *broader functionality*.

- *Stronger Security:* In the typical formulation of MIFE security, an attacker is allowed to either corrupt *all or none* of the users who can encrypt the data. In this work, we study MIFE security in a stronger and more natural model where we allow an attacker to corrupt any subset of the users, instead of only permitting all-or-nothing corruption. We formalize the model by providing each user a unique encryption key, and letting the attacker corrupt all non-trivial subsets of the encryption keys, while still maintaining the MIFE security for ciphertexts generated using honest keys. We construct a secure MIFE system for quadratic functions in this fine-grained corruption model from bilinear maps. Our construction departs significantly from the existing MIFE schemes as we need to tackle a more general class of attackers.
- *Broader Functionality:* The notion of multi-client functional encryption, MCFE, is a useful extension of MIFE. In MCFE, each encryptor can additionally tag each ciphertext with appropriate metadata such that ciphertexts with only matching metadata can be decrypted together. In more detail, each ciphertext is now annotated with a unique *label* such that ciphertexts encrypted for different slots can now only be combined together during decryption as long as the associated labels are an exact match for all individual ciphertexts. In this work, we upgrade our MIFE scheme to also support *ciphertext labelling*. While the functionality of our scheme matches that of MCFE for quadratic functions, our security guarantee falls short of

E. Kiltz and V. Vaikuntanathan (Eds.): TCC 2022, LNCS 13747, pp. 711–740, 2022.
https://doi.org/10.1007/978-3-031-22318-1_25

the general corruption model studied for MCFE. In our model, all encryptors share a secret key, therefore this yields a secret-key version of quadratic MCFE, which we denote by SK-MCFE. We leave the problem of proving security in the general corruption model as an important open problem.

1 Introduction

Functional encryption (FE) [18,19,31] is a generalization of public key encryption that enables fine grained control over access to encrypted data. In FE, the secret key is associated with a function f, the ciphertext is associated with an input $\mathbf{x}$ from the domain of f, and decryption enables recovery of $f(\mathbf{x})$ and nothing else. Importantly, no information about $\mathbf{x}$ is revealed beyond what is revealed by $\{f_i(\mathbf{x})\}_i$ for any set of secret decryption keys corresponding to functions $\{f_i\}_i$ in possession of the adversary. This *collusion resistance* property of FE makes it very suitable for computing on encrypted data – a ciphertext encrypting the genomic data of hundreds of individuals can now be decrypted using function keys corresponding to various statistical functionalities studying correlations between genomic sequences and disease, while guaranteeing privacy of individual genomic sequences. Motivated by several important applications, including the construction of the powerful notion of *indistinguishability obfuscation* (iO) [12,16], FE has received an enormous amount of attention in the community, with scores of elegant constructions from diverse assumptions, achieving various useful functionalities and satisfying assorted notions of security [3,6,14,15,20,24,25,34].

Multi-Input Functional Encryption. Functional encryption was first generalized to support aggregated computation over multiple input sources by the celebrated work of Goldwasser et al. [26]. The premise of multi-input FE, denoted by MIFE, is that in many natural applications of FE it is essential to support generalized functionalities where arity is greater than one. For instance, in the above example of genome wide association studies, the ciphertext must encrypt genomic data of multiple individuals for it to be useful for the statistical studies in question, but this suggests that this data must be encrypted all at once by a single entity, which is an unreasonable assumption in practice. Genomic data is highly sensitive information and it is much more meaningful to allow every individual to encrypt their own data locally and generalize the construction to support functions of large arity that can process several ciphertexts at a time. This constraint is organically captured by MIFE, where n independent encryptors may individually generate ciphertexts for vectors $\{\mathbf{x}_i^j\}_{i\in[n],j\in[\mathsf{poly}]}$ and a secret key for function f allows to compute $f(\mathbf{x}_1^{j_1},\mathbf{x}_2^{j_2},\ldots,\mathbf{x}_n^{j_n})$ for any $j_1,\ldots,j_n \in [\mathsf{poly}]$.

Since its inception, MIFE received substantial attention which quickly bifurcated into two parallel branches – (i) the first builds on top of powerful primitives such as iO or *compact* single-input FE for general models of computation, like circuits or Turing machines and uses these to construct MIFE for circuits or Turing machines [11,12,16], (ii) the second focuses on efficient direct constructions for restricted functionalities from simple assumptions such as pairings or learning with errors [1,2,4,5,9,21,23,29,32]. In this work, we continue development of the second branch by making advances to the recently proposed construction of MIFE for quadratic functions by Agrawal, Goyal, and Tomida [9].

Modelling Security. Given the tension between functionality and security, where functionality seeks to reveal partial information about the input, while security seeks to protect privacy of the input, the question of modelling security in functional encryption has turned out to be subtle, and has been examined in multiple works [7,8,18,30]. For the setting of *unbounded* collusion, namely where the adversary can obtain any polynomial number of function keys, in the security game, the *indistinguishability* based definition of security has emerged as the gold standard (due to impossibilities that plague the alternative simulation-based security [7,8,18]). In the single-input setting, both symmetric and public key FE have been studied and are relevant for different applications. In the multi-input setting, it was observed by Goldwasser et al. [26] that the symmetric key setting, where the encryptor requires a secret key to compute a ciphertext, is much more relevant for applications. This is to prevent the primitive from becoming meaningless due to excessive leakage occurring by virtue of functionality. In more detail, let us consider a two input scheme where a given first slot ciphertext hides a challenge bit b. Now, in the public key setting, an adversary can compute an unbounded encryptions for slot 2 herself and match these with the challenge ciphertext of slot 1 to learn a potentially unbounded amount of information. This unrestricted information leakage can be prevented by requiring the encryption algorithm to require a secret key.

However, in the symmetric key multi-input setting, an additional subtlety emerges related to the uniqueness of each user's encryption key. For instance, if we consider the application of encrypting genomic data discussed above, it quickly becomes apparent that having all users share the same encryption key is problematic – if the genomic data is encrypted and stored in a central repository, then any malicious insider, who has contributed data and is hence in possession of the master encryption key, can download and decrypt data belonging to any other user! As data is supposed to span hundreds of users, the master encryption key will become widely distributed and the privacy of honest user data can very quickly and easily get compromised. Hence, it is crucial for security that encryption keys be unique to users, and the adversary gaining control of a particular user's key does not compromise the security of other users' data.

Multi-Input FE for Quadratic Functions. Recently, Agrawal, Goyal, and Tomida (AGT) [9] provided the first construction of multi-input functional encryption for quadratic functions. In more detail, they construct an n-input MIFE scheme for the function class $\mathcal{F}_{m,n}$, which is defined as follows. Each function $f \in \mathcal{F}_{m,n}$ is represented by a vector $\mathbf{c} \in \mathbb{Z}^{(mn)^2}$. For inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{Z}^m$, f is defined as $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$ where $\mathbf{x} = (\mathbf{x}_1 || \cdots || \mathbf{x}_n)$ and $\otimes$ denotes the Kronecker product. In their quadratic MIFE scheme for $\mathcal{F}_{m,n}$, a user can encrypt $\mathbf{x}_i \in \mathbb{Z}^m$ to CT_i for slot $i \in [n]$, a key generator can compute a secret key SK for $\mathbf{c} \in \mathbb{Z}^{(mn)^2}$, and decryption of $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ with SK reveals only $\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle$ and nothing else.

However, while this result makes exciting progress in the domain of direct constructions for MIFE by providing the first candidate supporting quadratic functions, it suffers from the severe drawback that all the encryptors must share the same master key for encryption. As described above, this limits the applicability of the construction for many meaningful practical applications, e.g. when the system is susceptible to insider attacks. Moreover, having a single master key for all users creates a single point of failure which makes the system vulnerable to not only attack but also inadvertent leakage/misuse. Decentralizing trust is an overarching goal in cryptography, and this motivates to design a scheme where users have unique encryption keys and the adversarial model is strong enough to capture corruption of some subset of these.

Multi-Client Functional Encryption. A generalization of multi-input functional encryption is the notion of multi-client functional encryption (MCFE) where the ciphertext is additionally associated with a label. In more detail, encryptor i now encrypts not only the input $\mathbf{x}_i$ but also a public label ℓ_i to obtain $\mathsf{CT}(i, \mathbf{x}_i, \ell_i)$. A functional key SK_f for any n-ary function f can be used to decrypt $\{\mathsf{CT}(i, \mathbf{x}_i, \ell_i)\}_{i \in [n]}$ if and only if all the labels match, i.e. $\ell_i = \ell$ for all $i \in [n]$. Note that setting all labels to a single value (say "TRUE") recovers the notion of MIFE, which allows unrestricted combinations of ciphertexts across slots. The more expressive MCFE provides additional control over allowable combinations of ciphertexts, which is very useful for several applications – for instance, in the example of computing on encrypted genomic data discussed above, being able to filter records based on some label such as *ethnicity* = *African* may help to substantially reduce the number of inputs that participate in the study, making the process more efficient.

We emphasize that regardless of the security model (all-or-nothing or fine-grained), the motivation of labelling functionality is to better control the decryption pattern to reduce the information that a decrypter can learn. In the plain n input MIFE setting, where Q ciphertexts per slot are available, the decrypter can potentially compute Q^n function values, which reveal a large amount of information about the underlying plaintexts. However, using Q distinct labels to label every ciphertext in each slot, we can reduce the number of function values revealed to as little as Q. Thus, the labelling functionality is quite useful for controlling the amount of information that a decrypter learns.

It is worth noting that for an MIFE construction supporting general circuits, MCFE can easily be captured by adding an additional check in the function key to verify that all the labels are equal, but for restricted function classes like linear or quadratic functions, MCFE is more powerful than MIFE. In the arena of direct constructions from simple assumptions, the notion of MCFE has been studied for the case of linear functions [1,2,21,26,29] but not for quadratic functions, to the best of our knowledge.

Our Results. We advance the state-of-the-art in MIFE, and propose new constructions with *stronger security* and *broader functionality*.

- *Stronger Security:* Typically, in the MIFE security game, an attacker is allowed to either corrupt *all or none*[1] of the users who can encrypt the data. Here we study MIFE security in a "fine-grained" corruption model where an attacker can corrupt even *non-trivial* subset of the users, *instead of only the trivial subsets*.

  We formalize such a fine-grained corruption model by providing each user a unique encryption key, and letting the attacker corrupt any subset of the encryption keys. We require that, even after corruption of any non-trivial subset of encryption keys, the scheme still satisfies the MIFE-style security for all ciphertexts generated using honest encryption keys. We give a construction for a MIFE system whose security can be proven in this fine-grained corruption model, instead of the standard *all-or-nothing* corruption model. Our construction departs significantly from the existing AGT quadratic MIFE scheme [9] as we need to tackle a more general class of attackers.

  We observe that while several inner product MIFE schemes already have stronger security in the context of MCFE [1,10,22], achieving it in quadratic MIFE is much more difficult. Intuitively, a decrypter in a quadratic MIFE system is allowed to learn a function value on cross terms derived from different slots, and achieving this without heavy machinery such as obfuscation seems to require the encryption keys to be correlated with each other (this is also the case for the AGT scheme). Due to the correlation, the corruption of even a single encryption key affects the security of ciphertexts for all the other slots. This is in contrast to inner product MIFE, which is basically obtained by running independent single-input inner product FE instances in parallel.
- *Broader Functionality:* In MCFE, each encryptor can specify a special label, to tag each ciphertext with appropriate metadata, such that ciphertexts with only exactly matching metadata/labels can be decrypted together. Here we upgrade our MIFE scheme to additionally support *ciphertext labelling*. While the functionality of our upgraded MIFE scheme matches that of MCFE for quadratic functions, our security guarantee falls short of the general corruption model studied for MCFE. In our model, all encryptors share a secret key, therefore this yields a secret-key version of quadratic MCFE, which we denote by SK-MCFE. We leave the problem of proving security in the general corruption model as an important open problem.

1.1 Technical Overview

The starting point for both of our MIFE and SK-MCFE schemes for quadratic functions is the recent AGT scheme [9]. The AGT construction necessitates that

[1] An MIFE scheme where corruption of all encrypting users is allowed is more commonly regarded as public-key MIFE, while disallowing corruption of any encrypting user is regarded as secret-key MIFE.

all encryptors share the same master secret key, thus throughout the sequel we will refer to it as the "SK-MIFE" scheme.

A Simplified Overview of the AGT SK-MIFE Scheme. The AGT scheme uses three building blocks – (i) SK-FE for inner product (IPFE), (ii) SK-FE for *predicate* inner product (pIPFE), and (iii) SK-MIFE for *mixed-group* inner product. The mixed-group property of (iii) is necessary for a technical reason in the security proof, but for now we can consider it as SK-MIFE for inner product (IP-MIFE). And, for security, all of the underlying schemes are required to satisfy the corresponding function-hiding security property. Concretely, the required MIFE schemes are summarized in Table 1.[2]

Table 1. Description of input and function classes for IPFE, pIPFE, IP-MIFE.

Scheme type	No. of inputs	Input class(es)	Function class	Description of functions
IPFE	1	$\mathcal{X} = \mathbb{Z}_p^m$	$\mathcal{F} = \mathbb{Z}_p^m$	$f_{\mathbf{y}}(\mathbf{x}) = \langle \mathbf{x}, \mathbf{y} \rangle$
pIPFE	1	$\mathcal{X} = \mathbb{Z}_p^{m_1} \times \mathbb{Z}_p^{m_2}$	$\mathcal{F} = \mathbb{Z}_p^{m_1} \times \mathbb{Z}_p^{m_2}$	$f_{\mathbf{y}_1,\mathbf{y}_2}(\mathbf{x}_1, \mathbf{x}_2) = \begin{cases} \langle \mathbf{x}_2, \mathbf{y}_2 \rangle & \text{if } \langle \mathbf{x}_1, \mathbf{y}_1 \rangle = 0, \\ \perp & \text{otherwise.} \end{cases}$
IP-MIFE	n	$\mathcal{X}_1 = \cdots \mathcal{X}_n = \mathbb{Z}_p^m$	$\mathcal{F} = \mathbb{Z}_p^{mn}$	$f_{\mathbf{y}}(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \langle (\mathbf{x}_1 \| \ldots \| \mathbf{x}_n), \mathbf{y} \rangle$

Notation. We denote IPFE ciphertexts of $\mathbf{v}$ by $\mathsf{iCT}[\mathbf{v}]$, pIPFE ciphertexts of $(\mathbf{v}_1, \mathbf{v}_2)$ by $\mathsf{pCT}[(\mathbf{v}_1, \mathbf{v}_2)]$ and IP-MIFE ciphertexts of $\mathbf{v}$ for slot i by $\mathsf{miCT}_i[\mathbf{v}]$ under some master secret keys iMSK, pMSK, miMSK, respectively. Similarly we denote IPFE secret keys of $\mathbf{v}$ by $\mathsf{iSK}[\mathbf{v}]$, pIPFE secret keys of $(\mathbf{v}_1, \mathbf{v}_2)$ by $\mathsf{pSK}[(\mathbf{v}_1, \mathbf{v}_2)]$ and IP-MIFE secret keys for $\mathbf{v}$ by $\mathsf{miSK}[\mathbf{v}]$ under the same master secret keys iMSK, pMSK, miMSK, respectively.

AGT Scheme Description. Let us start by recalling the structure of ciphertexts and secret keys in the AGT SK-MIFE scheme. At a high level, an AGT ciphertext CT_i of $\mathbf{x} \in \mathbb{Z}^m$ and SK for $\mathbf{c} \in \mathbb{Z}_p^{(mn)^2}$ are of the following form:

$$\mathsf{CT}_i = \left(\left\{ \mathsf{pCT}[(\mathbf{h}, \mathbf{b}_j)],\ \mathsf{pSK}[(\widetilde{\mathbf{h}}, \widetilde{\mathbf{b}}_j)] \right\}_{j \in [m]},\ \mathsf{iCT}[\mathbf{d}],\ \mathsf{iSK}[\widetilde{\mathbf{d}}],\ \mathsf{miCT}_i[\mathbf{f}] \right) \tag{1}$$

$$\mathsf{SK} = \left(\{\sigma_{i,k}\}_{i,k \in [n]}, \mathsf{miSK}[\widetilde{\mathbf{f}}] \right) \tag{2}$$

for some $\mathbb{Z}_p$ vectors $\mathbf{b}_j, \widetilde{\mathbf{b}}_j, \mathbf{d}, \widetilde{\mathbf{d}}, \mathbf{f}, \widetilde{\mathbf{f}}, \mathbf{h}, \widetilde{\mathbf{h}}$ and $\mathbb{Z}_p$ elements $\sigma_{i,k}$.

Now a message vector $\mathbf{x}$ is encoded in the vectors $\mathbf{b}_j, \widetilde{\mathbf{b}}_j$, and the remaining vectors in the ciphertext are only added to either tie together separate components of different AGT ciphertexts, or randomize a portion of a single AGT ciphertext. We refer the reader to [9] for a more detailed overview, but for our purposes, it is enough to understand how the decryption algorithm works.

[2] Formally, the inner product functionalities defined need to involve group elements as it is necessary for the proof. However, for simplicity of the overview, we use directly define them over $\mathbb{Z}_p$.

Consider a sequence of n AGT ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ and a corresponding secret key SK. The decryptor first runs the decryption algorithm for the pIPFE scheme for all possible input combinations. That is, for all $i, k \in [n]$ and $j, \ell \in [m]$, it computes

$$z_{i,j,k,\ell} = \mathsf{pDec}(\mathsf{pCT}[(\mathbf{h}_i, \mathbf{b}_{i,j})], \mathsf{pSK}[(\widetilde{\mathbf{h}}_k, \widetilde{\mathbf{b}}_{k,\ell})]). \quad (3)$$

As it turns out, the underlying encoding procedure used in AGT ensures that each such term is of the form $z_{i,j,k,\ell} = \mathbf{x}_i[j]\mathbf{x}_k[\ell] + u_{i,j,k,\ell}$, where $u_{i,j,k,\ell}$ is a pseudorandom masking term such that $\sum \mathbf{c}[(i,j,k,\ell)]u_{i,j,k,\ell} = \langle \mathbf{c}, \mathbf{u} \rangle$ can be computed by combining the remaining portions of the ciphertexts and secret key. That is, the decryptor first computes

$$\sum \mathbf{c}[(i,j,k,\ell)]z_{i,j,k,\ell} = \sum \mathbf{c}[(i,j,k,\ell)]\mathbf{x}_i[j]\mathbf{x}_k[\ell] + \sum \mathbf{c}[(i,j,k,\ell)]u_{i,j,k,\ell}$$

where $\sum \mathbf{c}[(i,j,k,\ell)]\mathbf{x}_i[j]\mathbf{x}_k[\ell]$ is the desired output, and then it computes $\sum \mathbf{c}[(i,j,k,\ell)]u_{i,j,k,\ell} = \langle \mathbf{c}, \mathbf{u} \rangle$ by combining the $(\mathsf{iCT}[\mathbf{d}], \mathsf{iSK}[\widetilde{\mathbf{d}}], \mathsf{miCT}_i[\mathbf{f}])$ portion of each ciphertext amongst themselves and also with the secret key $(\{\sigma_{i,k}\}_{i,k\in[n]}, \mathsf{miSK}[\widetilde{\mathbf{f}}])$.

Achieving Strong Fine-Grained Security. Recall that in the stronger fine-grained corruption model, each encryptor has a unique encryption key, and the adversary is allowed to corrupt any subset of encryption keys in the security game. Throughout the sequel, we refer to such a scheme as plain MIFE in contrast to SK-MIFE.

Before describing our main ideas, we highlight the reason as to why AGT is not already secure in this stronger corruption model. Observe that each component of the AGT ciphertext CT_i is generated under the same master secret key of the corresponding scheme over all slots. In other words, it is essential that all encryption keys include the same IPFE, pIPFE, and IP-MIFE master secret keys. As it turns out, this is one of the main barriers to proving the SK-MIFE construction of AGT to be strongly secure. This is because the scheme ends up being completely insecure if encryption keys for any slot are revealed! Basically, revealing only the underlying pIPFE master secret key allows one to completely decrypt any ciphertexts of the AGT scheme.

While this seems like a major technical barrier at first, we observe that there is a very elegant way to get around this problem by relying on the underlying homomorphic properties satisfied by the SK-MIFE scheme. Although, the AGT SK-MIFE construction can not be used as is since the usage of the pIPFE scheme prevents any useful type of ciphertext homomorphism, we are able to simplify the underlying SK-MIFE construction that not only avoids the usage of pIPFE completely, but also leads to an interesting homomorphism property that we show is very useful in upgrading any weakly secure SK-MIFE into a strongly secure MIFE scheme.

The Special Property. Let us start by describing the special homomorphism property P that we crucially rely on. It states that there exists an *explicit* and *efficient* algorithm $\widetilde{\mathsf{Enc}}$, and a sequence of *public* elementary messages $e_{i,1}, \ldots, e_{i,d} \in \mathcal{X}_i$ ($\forall i \in [n]$) such that – for every slot $i \in [n]$ and message $x_i \in \mathcal{X}_i$, the following two distributions are statistically indistinguishable:

$$\left\{ \left(\mathsf{PP}, \{\mathsf{CT}_{i,j}\}_{j \in [d]}, \mathsf{CT}_i\right) : \mathsf{CT}_i \leftarrow \mathsf{Enc}(\mathsf{MSK}, i, x_i) \right\},$$
$$\left\{ \left(\mathsf{PP}, \{\mathsf{CT}_{i,j}\}_{j \in [d]}, \mathsf{CT}_i\right) : \mathsf{CT}_i \leftarrow \widetilde{\mathsf{Enc}}\left(\{\mathsf{CT}_{i,j}\}_j, x_i\right) \right\}$$

where $(\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathsf{CT}_{i,j} \leftarrow \mathsf{Enc}(\mathsf{MSK}, i, e_{i,j})$ for $j \in [d]$.

Property P to MIFE. Assuming there exists an SK-MIFE scheme satisfying property P, our main observation is that there exists a generic compiler to upgrade it to a MIFE for the same function class in which an attacker can corrupt any arbitrary set of encryption keys. That is, consider any SK-MIFE scheme $(\mathsf{Setup}', \mathsf{Enc}', \mathsf{KeyGen}', \mathsf{Dec}')$ for some function class $\mathcal{F}$ satisfying property P, our compiler upgrades it to an MIFE scheme $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{KeyGen}', \mathsf{Dec}')$ for $\mathcal{F}$ as follows:

$\mathsf{Setup}(1^\lambda, 1^n)$: It computes $\mathsf{PP}, \mathsf{MSK} \leftarrow \mathsf{Setup}'(1^\lambda)$ and $\mathsf{CT}_{i,j} \leftarrow \mathsf{Enc}'(\mathsf{MSK}, i, e_{i,j})$ for all $i \in [n], j \in [d]$, and sets $\mathsf{EK}_i = \{\mathsf{CT}_{i,j}\}_j$ for all $i \in [n]$. Then, it outputs the parameters as $\mathsf{PP}, \{\mathsf{EK}_i\}_i, \mathsf{MSK}$.

$\mathsf{Enc}(\mathsf{EK}_i, x)$: It computes $\mathsf{CT}_i \leftarrow \widetilde{\mathsf{Enc}}\left(\{\mathsf{CT}_{i,j}\}_j, x\right)$ and outputs CT_i.

The correctness follows directly from the correctness of the underlying SK-MIFE scheme and the statistical closeness of the output distributions between Enc and $\widetilde{\mathsf{Enc}}$. And, the proof of security also follows via a hybrid argument. The main idea is to first switch how each challenge ciphertext is generated. That is, instead of computing it as $\widetilde{\mathsf{Enc}}\left(\{\mathsf{CT}_{i,j}\}_j, x^\beta\right)$, the challenger computes it directly as $\mathsf{Enc}(\mathsf{MSK}, i, x^\beta)$ (where $\beta \in \{0, 1\}$ and x^0, x^1 are the challenge messages). Note that this readily follows from the statistical closeness, and thus, by relying on the regular security of the underlying SK-MIFE scheme, we can prove the stronger security for our MIFE scheme. This is because the reduction algorithm can simulate a corrupted encryption key $\mathsf{EK}_i = \{\mathsf{CT}_{i,j}\}_{j \in [d]}$ by querying its own oracle on the elementary messages $e_{i,1}, \ldots, e_{i,d}$. For more details, we refer the reader to the main body.

Building SK-MIFE with Property P. In order to obtain our final result, we need to instantiate the above generic compiler with an SK-MIFE scheme for quadratic functions with property P. As mentioned earlier, our core idea in this part is to rely on the homomorphic structure of the AGT SK-MIFE scheme. Recall that a ciphertext in the AGT scheme consists of bilinear source group elements. Thus, we can define a group operation over the AGT ciphertexts by element-wise multiplication of group elements (and we use addition for the group operation in what follows). Let $\mathsf{CT}_i[\mathbf{x}]$ be a slot-i encryption of $\mathbf{x}$ in the AGT scheme. Our observation is that if for any $a_1, a_2 \in \mathbb{Z}_p$, we have

$$a_1 \mathsf{CT}_i[\mathbf{x}_1] + a_2 \mathsf{CT}_i[\mathbf{x}_2] = \mathsf{CT}_i[a_1 \mathbf{x}_1 + a_2 \mathbf{x}_2], \tag{4}$$

then we can achieve P by simply setting the elementary messages to be $\mathbf{e}_1, \ldots, \mathbf{e}_n$, where $\mathbf{e}_j$ is the one-hot vector with the j-th element being one, and defining $\widetilde{\mathsf{Enc}}$ using the appropriate group operations. Unfortunately, this is not the case!

Insufficiency of AGT. To better understand the reason for failure, we need to open up the encryption abstractions used in AGT to their underlying bilinear form. Informally, an AGT ciphertext CT_i for $\mathbf{x} \in \mathbb{Z}_p^m$ looks like $\mathsf{CT}_i = ([\mathbf{v}\mathbf{M}_i]_1, [\mathbf{w}\mathbf{N}_i]_2)$. Here $[\cdot]_1, [\cdot]_2$ denote element-wise group exponentiation in bilinear groups G_1, G_2, and $\mathbf{M}_i, \mathbf{N}_i$ are common matrices shared among all ciphertexts for slot i. Also, each element of $\mathbf{v}, \mathbf{w}$ depends on $\mathbf{x}$ and the random tape used in encryption. Concretely, each element of $\mathbf{v}, \mathbf{w}$ is one of the following four types — (i) 1; (ii) $\mathbf{x}[j]$ for some $j \in [m]$; (iii) a fresh random $\mathbb{Z}_p$ element; or (iv) an element of the tuple $(b, c, b\ell, c\ell)$ where b, c, ℓ are fresh random $\mathbb{Z}_p$ elements.

From the viewpoint of well-formedness of a homomorphically operated ciphertext, it is not hard to see that the elements (ii) and (iii) will stay consistent with the homomorphism Eq. (4), while the elements (i) and (iv) will no longer be well-formed after the group operations. This is because, after the homomorphic addition as Eq. (4), the element (i) becomes $a_1 + a_2$, while the element (iv) become an elements of the tuple $(a_1b_1 + a_2b_2, a_1c_1 + a_2c_2, a_1b_1\ell_1 + a_2b_2\ell_2, a_1c_1\ell_1 + a_2c_2\ell_2)$. While an element (i) can still be well-formed as long as $a_1 + a_2 = 1$, an element (iv) will never be well-formed (unless $\ell_1 = \ell_2$, which occurs with only negligible probability).

Stripping Away pIPFE from AGT. Diving a bit further into the structure and semantics of the AGT SK-MIFE scheme, we find out that the elements (iv) are derived from the pIPFE scheme. So, a natural thought is if we can remove the pIPFE scheme from AGT, then we can eliminate the elements (iv) thereby solving the above problem. However, the usage of the pIPFE scheme in the AGT template was crucial as replacing it with a (non-predicate) IPFE scheme enabled a mix-and-match attack wherein an attacker can illegally combine portions of two different ciphertexts for the *same slot*. Concretely, for two ciphertexts $\mathsf{CT}_i^1, \mathsf{CT}_i^2$ in the same slot, pIPFE prevents decryptor from computing $\mathsf{pDec}(\mathsf{pCT}_{i,j}^1, \mathsf{pCT}_{i,\ell}^2)$ in the decryption process as in Eq. (3) (meaning that $\langle \mathbf{h}^1, \widetilde{\mathbf{h}}^2 \rangle \neq 0$ if $\mathbf{h}^1$ and $\widetilde{\mathbf{h}}^2$ are vectors derived from two different ciphertexts for the same slot i).

Although this seems to be a major bottleneck at first, we make an important observation that if each encryptor computes and encrypts all possible quadratic terms between its own message vector at the time of encryption, then a decryptor does not need to generate the quadratic terms derived from the same slot via the pIPFE decryption. Therefore, the mix-and-match problem can be rather easily solved by replacing pIPFE with a plain (non-predicate) IPFE scheme. And, since this new encryption method only increases the length of the underlying encrypted vector from m to m^2, thus it is still efficient. We refer to Definition 2.6 and 2.7 for more details.

Final Rerandomization Trick. While it seems that we are done at this point, unfortunately this is still not sufficient. And, the reason is the fact that even after removing elements (iv), we cannot achieve the property P by using $\mathbf{e}_1, \ldots, \mathbf{e}_n$ as the public elementary messages from two reasons. First, $\sum_j \mathbf{x}[j]$ is not necessarily 1, and thus elements (i) may not be 1 after the homomorphic addition. Second, elements (iii) depend on $\mathbf{x}$ and the random tape used to generate the ciphertexts of $\mathbf{e}_i$, and thus not independently random after the homomorphic addition. The second reason can be visualized as the resulting ciphertext containing far less entropy than a freshly sampled ciphertext.

However, we solve these issues by the following rerandomization trick. Our idea is to additionally include a large sequence of $\mathbf{0}$ vectors to the list of elementary messages, and sample a fresh sequence of random elements which will be used to homomorphically add each encryption of $\mathbf{0}$ to the underlying homomorphically computed ciphertext such that the resulting ciphertext has sufficient entropy. That is, for a sufficiently large D, we define $\widetilde{\mathsf{Enc}}$ as follows: $\widetilde{\mathsf{Enc}}\left((\{\mathsf{CT}_i[\mathbf{e}_j]\}_{j\in[m]}, \{\mathsf{CT}_{i,j}[\mathbf{0}]\}_{j\in[D]}), \mathbf{x}\right)$ computes $\mathsf{CT}_i[\mathbf{x}]$ as

$$\mathsf{CT}_i[\mathbf{x}] = \mathsf{CT}_{i,1}[\mathbf{0}] + \sum_{j\in[m]} \mathbf{x}_i[j](\mathsf{CT}_i[\mathbf{e}_j] - \mathsf{CT}_{i,1}[\mathbf{0}]) + \sum_{j\in[\frac{D-1}{2}]} \gamma_j(\mathsf{CT}_{i,2j}[\mathbf{0}] - \mathsf{CT}_{i,2j+1}[\mathbf{0}]),$$

where $\gamma_1, \ldots, \gamma_{(D-1)/2} \leftarrow \mathbb{Z}_p$.

This solves the second problem as now the elements (iii) are distributed randomly if D is sufficiently large due to the fresh entropy introduced by $\gamma_1, \ldots, \gamma_{(D-1)/2}$. And, since we have $\sum_{j\in[m]}(\mathbf{x}_i[j] - \mathbf{x}_i[j]) + \sum_{j\in[(D-1)/2]}(\gamma_j - \gamma_j) = 0$, thus element (i) is also equal to 1 in $\mathsf{CT}_i[\mathbf{x}]$. Hence, the above rerandomization trick combined with the pIPFE removal strategy gives us our SK-MIFE scheme for quadratic functions with property P, which in turn gives us our quadratic MIFE scheme secure in the stronger fine-grained corruption model.

Supporting the Ciphertext Labelling Functionality. Finally, we provide a rather simple yet incredibly useful mechanism to annotate labels with SK-MIFE ciphertexts. This adds the feature of multi-client style encryption to the quadratic SK-MIFE scheme. To this end, we look back at the existing techniques to achieve desired labelling for IP-MIFE schemes (that is, the ideas used to obtain IP-MCFE, or in other words, MCFE for inner product), but find that all techniques are rather specific to inner product. The prior works basically use the following blueprint [1,10,21,22]. The MCFE schemes use a (single-input) IPFE scheme as a building block, and a ciphertext of the MCFE for the i-th slot message $\mathbf{x}_i$ with a label lab is simply a ciphertext of the IPFE scheme for some vector $\widetilde{\mathbf{x}}_i$ related to $\mathbf{x}_i$ and lab. A secret key of the MCFE scheme for $\mathbf{c} = (\mathbf{c}_1 || \ldots || \mathbf{c}_n)$ contains IPFE secret keys for some vector $\widetilde{\mathbf{c}}_i$ related to $\mathbf{c}$ for $i \in [n]$, and decryption for slot-i reveals

$$\langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{c}}_i \rangle = \langle \mathbf{x}_i, \mathbf{c}_i \rangle + u_i$$

where u_i is a masking term such that $\sum_{i\in[n]} u_i$ is equal to 0 (or a computable value by the decryptor) only when $\widetilde{\mathbf{x}}_i$ is associated with the same label for all i. Hence, the decryptor can learn only $\sum_i \langle \mathbf{x}_i, \mathbf{c}_i \rangle$ as desired. However, the structure of the only known MIFE scheme for quadratic functions by AGT, as observed, is quite different from this blueprint, and thus we need a new approach.

Our starting point is again the AGT MIFE scheme where recall the ciphertext has the form as described in Eq. (1). A natural first thought is to try to replace all the three underlying IPFE, pIPFE, and IP-MIFE schemes with their labelled counterparts. After a quick glance, it appears that this would be a viable strategy since if we could annotate each component in the AGT ciphertext with a label, then the entire AGT ciphertext will be labelled as well.

As we elaborated during the description of our MIFE construction, the application of pIPFE in the AGT template can be replaced with any IPFE scheme (ignoring the quadratic increase in the overall ciphertext size). Concretely, we showed that the ciphertext CT of the modified AGT scheme can be written as

$$\mathsf{CT}_i = \left(\left\{ \mathsf{iCT}^{(1)}[\mathbf{b}_j], \mathsf{iSK}^{(1)}[\widetilde{\mathbf{b}}_j] \right\}_{j \in [m]},\ \mathsf{iCT}^{(2)}[\mathbf{d}],\ \mathsf{iSK}^{(2)}[\widetilde{\mathbf{d}}],\ \mathsf{miCT}_i[\mathbf{f}] \right),$$

where $(\mathsf{iCT}^{(1)}, \mathsf{iSK}^{(1)})$ and $(\mathsf{iCT}^{(2)}, \mathsf{iSK}^{(2)})$ are generated by two separate master secret keys $\mathsf{iMSK}^{(1)}$ and $\mathsf{iMSK}^{(2)}$, respectively. Thus, it seems like if we can annotate both, the IPFE and the IP-MIFE, components of the modified AGT scheme with the same label, then the resulting quadratic MIFE scheme will also support ciphertext labelling functionality.

Now to annotate the IP-MIFE component of the ciphertext, we need a labelled version of the SK-MIFE scheme for *mixed-group* inner products with *function-hiding security* as a counterpart. Although such a scheme for inner product is not already known, we were able to construct a new scheme with the desired properties by combining ideas from the SK-MIFE scheme for mixed-group inner product in [9] and the MCFE scheme for inner product in [10]. We refer the reader to Sect. 3 for the exact details.

Finally, to get the desired result, we simply need a mechanism to annotate the IPFE component of the AGT ciphertexts with labels such that ciphertexts with different labels can no longer be combined. Our idea is to simply keep a PRF key K as part of the overall system master key, and use the PRF key K to sample a *label-dependent* IPFE key at the time of encryption. That is, the setup no longer samples the IPFE keys used during the encryption, but instead the encryptor first samples the IPFE keys using $\mathsf{PRF}(K, \mathsf{lab})$ as the randomness where lab is the specified label, and then uses those keys to compute the appropriate ciphertext components. Clearly, ciphertexts encrypted w.r.t. different labels can no longer be combined since the underlying ciphertext components are now incompatible (as they are sampled using independent IPFE keys). And, basically by iterating the hybrid sequence of the SK-MIFE scheme for quadratic functions in [9] *per queried label*, we can also prove security in the secret-key MCFE setting.

Open Problems. We conclude the introduction by discussing some open problems. To the best of our knowledge, this is the first work proposing a technique to convert SK-MIFE to MIFE with stronger security. Since our technique is applicable to all SK-MIFE schemes with property P, exploring other classes of MIFE to which our technique is applicable is an interesting open problem. We observe that this conversion does seem applicable to group-based SK-MIFE schemes for inner product in [4,5] since they enjoy a nice homomorphic property. However, MIFE schemes for inner product with the stronger security are already known so this does not yield a new result. Nevertheless it does give a new pathway to obtaining these results since known MCFE schemes for inner product are constructed without going through SK-MIFE.

The second open question is the construction of a (public-key) MCFE scheme for quadratic functions. Interestingly, while the above ideas are sufficient for SK-MCFE for quadratic functions, we were unable to prove security in the public-key setting. First, in the above abstraction, the usage of PRFs to annotate the IPFE portion of the modified AGT ciphertext requires the encryption key for each slot to contain the secret PRF key K. Thus, corruption of even one encryption key completely breaks down the scheme. An approach is to sample a separate PRF key for each pair of encryption slots, however, even that does not seem to suffice as corrupting even a single secret key for a particular encryption slot seems to provide an attacker a mechanism to maul the labels from honest ciphertexts, thereby breaking security. Other natural approaches run into similar roadblocks. We leave the question full fledged MCFE as an exciting open problem.

We remark that the approach of providing generic compilers to "upgrade" security notions of primitives can be very useful in enabling new constructions since it simplifies the minimum building block that must be instantiated. For the case of restricted functionalities like linear [1] or quadratic functions (this), such compilers have required the underlying scheme to satisfy "nice" algebraic properties. Can this requirement be removed? Given current techniques, it seems difficult to remove such requirements without relying on strong tools like obfuscation. However, exploring this question more fully is a promising line of research.

Finally, it is evidently a fascinating question whether we can "lift" the degree of the underlying function class beyond 2 without relying on strong tools like compact functional encryption or obfuscation. Currently, we have results from single assumptions in the arena of degree ≤ 2 [1,2,4,5,9,21,23,29,32] and results from combinations of assumptions for classes like NC_1 and beyond [27,28] even in the single input setting. While compact functional encryption can be generalized to the multi-input setting [13,17], can we have constructions of MIFE and MCFE for bigger classes of functions without relying on obfustopia primitives?

2 Preliminaries

Notation. We begin by defining the notation that we will use throughout the paper. We use bold letters to denote vectors and the notation $[a, b]$ to denote the set of integers $\{k \in \mathbb{N} \mid a \leq k \leq b\}$. We use $[n]$ to denote the set $[1, n]$. For vector $\mathbf{v}$, $\mathbf{v}[i]$ denotes the i-th element of $\mathbf{v}$. For $(i_n, \ldots, i_1) \in [N_n] \times \cdots \times [N_1] \subset \mathbb{N}^n$, we sometimes identify $(i_n, \ldots, i_1)$ as $\sum_{j\in[2,n]} \Big((i_j - 1)\prod_{\ell\in[j-1]} N_\ell\Big) + i_1$, which is an element in $[N_1 N_2 \cdots N_n]$. This identification is used to introduce an order in the elements in $[N_1] \times \cdots \times [N_n]$. For a matrix $\mathbf{A} = (a_{j,\ell})_{j,\ell}$ over $\mathbb{Z}_p$, $[\mathbf{A}]_i$ denotes a matrix over G_i whose (j, ℓ)-th entry is $g_i^{a_{j,\ell}}$, and we use this notation for vectors and scalars similarly. Throughout the paper, we use λ to denote the security parameter.

We will use a pseudorandom function (PRF) and standard cryptographic bilinear groups where the matrix decisional Diffie-Hellman (MDDH) assumption holds.

2.1 Multi-Input Functional Encryption

Syntax. Let n be the number of encryption slots, and $\mathcal{F} = \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ be a function family such that, for all $f \in \mathcal{F}_n$, $f : \mathcal{X}_1 \times \cdots \times \mathcal{X}_n \to \mathcal{Y}$. Here $\mathcal{X}_i$ and $\mathcal{Y}$ be the input and output spaces (respectively). A multi-input functional encryption (MIFE)[3] scheme for function family $\mathcal{F}$ consists of following algorithms.

$\mathsf{Setup}(1^\lambda, 1^n) \to (\mathsf{PP}, \{\mathsf{EK}_i\}_i, \mathsf{MSK})$. It takes a security parameter 1^λ, number of slots 1^n, and outputs public parameters PP, n encryption keys $\{\mathsf{EK}_i\}_{i\in[n]}$, a master secret key MSK. (The remaining algorithms implicitly take PP as input.)

$\mathsf{Enc}(\mathsf{EK}_i, x) \to \mathsf{CT}_i$. It takes the i-th encryption key EK_i and an input $x \in \mathcal{X}_i$, and outputs a ciphertext CT_i.

$\mathsf{KeyGen}(\mathsf{MSK}, f) \to \mathsf{SK}$. It takes the master key MSK and a function $f \in \mathcal{F}$ as inputs, and outputs a decryption key SK.

$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK}) \to y$. It takes n ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ and decryption key SK, and outputs a decryption value $y \in \mathcal{Y}$ or a special abort symbol $\perp$.

Correctness. An MIFE scheme for function family $\mathcal{F}$ is correct if for all $\lambda, n \in \mathbb{N}$, $(x_1, \ldots, x_n) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, $f \in \mathcal{F}_n$, we have

$$\Pr\left[y = f(x_1, \ldots, x_n) : \begin{array}{l} (\mathsf{PP}, \{\mathsf{EK}_i\}_i, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \{\mathsf{CT}_i \leftarrow \mathsf{Enc}(i, \mathsf{EK}_i, x_i)\}_i \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \\ y = \mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK}) \end{array} \right] = 1.$$

Definition 2.1. *For security, we define two indistinguishability-based security definitions: message-hiding security and function-hiding security. An MIFE scheme is sel-XX-YY-IND-secure (XX $\in \{pos, any\}$, YY $\in \{mh, fh\}$)*[4] *if for any stateful* admissible *PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(n)(\cdot)$ *such that for all $\lambda, n \in \mathbb{N}$, the following probability is negligibly close to $1/2$ in λ:*

[3] When $n = 1$, we call MIFE just functional encryption (FE).

[4] "sel" stands for "selective" meaning that the adversary has to select the challenge elements at the beginning of the security game. The opposite notion is "adaptive". "pos" stands for "positive". In MCFE, a user can decrypt ciphertexts only when it has ciphertexts for all slots with the same label, and a portion of them is useless for decryption. "pos" prohibits the adversary from querying the oracle on such useless challenge elements. "mh" and "fh" stand for "message-hiding" and "function-hiding", respectively.

$$\Pr\left[\mathcal{A}(\{\mathsf{EK}_i\}_{i\in\mathcal{CS}}, \{\mathsf{CT}_\mu\}_\mu, \{\mathsf{SK}_\nu\}_\nu) = \beta : \begin{array}{l} \beta \leftarrow \{0,1\} \\ \mathsf{PP}, \{\mathsf{EK}_i\}_{i\in[n]}, \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ (\mathcal{CS},\mathcal{MS},\mathcal{FS}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{PP}) \ \ s.t. \\ \quad \mathcal{CS}\subseteq[n] \\ \quad \mathcal{MS} = \{i^\mu, x^{\mu,0}, x^{\mu,1}\}_{\mu\in[q_c]} \\ \quad \mathcal{FS} = \{f^{\nu,0}, f^{\nu,1}\}_{\nu\in[q_k]} \\ \{\mathsf{CT}_\mu \leftarrow \mathsf{Enc}(i^\mu, \mathsf{EK}_{i^\mu}, x^{\mu,\beta})\}_\mu \\ \{\mathsf{SK}_\nu \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{\nu,\beta})\}_\nu \end{array}\right]$$

where the adversary $\mathcal{A}$ is said to be admissible if and only if:

1. $f^0(x_1^0, \ldots, x_n^0) = f^1(x_1^1, \ldots, x_n^1)$ *for all sequences* $(x_1^0, \ldots, x_n^0, x_1^1, \ldots, x_n^1, f^0, f^1)$ *such that:*
   - *For all* $i \in [n]$, $[(i, x_i^0, x_i^1) \in \mathcal{MS}]$ *or* $[i \in \mathcal{CS}$ *and* $x_i^0 = x_i^1]$,
   - $(f^0, f^1) \in \mathcal{FS}$.
2. *When XX = pos,* $q_c[i] > 0$ *for all* $i \in [n]$, *where* $q_c[i]$ *denotes the number of elements of the form* $(i, *, *)$ *in* $\mathcal{MS}$.
3. *When YY = mh,* $f^{\nu,0} = f^{\nu,1}$ *for all* $\nu \in [q_k]$.

MIFE security in secret-key setting. We say an MIFE scheme is secret-key MIFE (SK-MIFE) scheme if all the n encryption keys EK_i are basically the master secret key MSK. The security of an SK-MIFE scheme is defined the same way as an MIFE scheme except that the adversary has to set $\mathcal{CS} = \emptyset$.

2.2 Multi-Client Functional Encryption

A multi-client functional encryption (MCFE) scheme is an extension of MIFE where each ciphertext is now annotated with a unique label such that ciphertexts encrypted for different slots can now only be combined together during decryption as long as the associated labels match for all individual ciphertext pieces. We first define its syntax where we highlight in terms of changes, how MCFE compares with MIFE.

Syntax. An MCFE system is associated with a label space $\mathcal{L}$, in addition to the number of encryption slots n and function class $\mathcal{F}$ as in MIFE. A multi-client functional encryption scheme for function family $\mathcal{F}$ consists of following algorithms.

$\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Dec}$ have the same syntax as in MIFE.

$\mathsf{Enc}(\mathsf{EK}_i, \mathsf{lab}, x) \rightarrow \mathsf{CT}$. The encryption algorithm takes the i-th encryption key EK_i, a label lab, and an input $x \in \mathcal{X}_i$, and outputs a ciphertext CT_i.

Correctness. An MCFE scheme for function family $\mathcal{F}$ is correct if for all $\lambda, n \in \mathbb{N}$, $(x_1, \ldots, x_n) \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_n$, $f \in \mathcal{F}$, and label $\mathsf{lab} \in \mathcal{L}$, we have

$$\Pr\left[y = f(x_1, \ldots, x_n) : \begin{array}{l} (\mathsf{PP}, \{\mathsf{EK}_i\}_i, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n) \\ \{\mathsf{CT}_i \leftarrow \mathsf{Enc}(i, \mathsf{EK}_i, \mathsf{lab}, x_i)\}_i \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \\ y = \mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK}) \end{array}\right] = 1.$$

That is, if all the ciphertexts are encrypted for the same label, then the decryption works as in MIFE.

MCFE Security in Secret-key Setting. In this work we are mostly interested in the secret-key setting. The intuition behind security for secret-key MCFE is similar to that for secret-key MIFE, with the difference that the admissibility constraint for ciphertexts is defined for each label individually. Below we define it formally.

Definition 2.2. *An SK-MCFE scheme is sel-XX-YY-IND-secure (XX $\in$ {pos, any}], YY $\in$ {mh, fh}) if for any stateful* admissible *PPT adversary $\mathcal{A}$, there exists a negligible function* $\mathsf{negl}(n)(\cdot)$ *such that for all $\lambda, n \in \mathbb{N}$, the following probability is negligibly close to $1/2$ in λ:*

$$\Pr\left[\mathcal{A}(\{\mathsf{CT}_\mu\}_\mu, \{\mathsf{SK}_\nu\}_\nu) = \beta : \begin{array}{l} (\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n), \beta \leftarrow \{0,1\} \\ (\mathcal{MS}, \mathcal{FS}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{PP}) \ \ s.t. \\ \quad \mathcal{MS} = \{i^\mu, \mathsf{lab}^\mu, x^{\mu,0}, x^{\mu,1}\}_{\mu \in [q_c]} \\ \quad \mathcal{FS} = \{f^{\nu,0}, f^{\nu,1}\}_{\nu \in [q_k]} \\ \{\mathsf{CT}_\mu \leftarrow \mathsf{Enc}(\mathsf{MSK}, i^\mu, \mathsf{lab}^\mu, x^{\mu,\beta})\}_\mu \\ \{\mathsf{SK}_\nu \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f^{\nu,\beta})\}_\nu \end{array}\right]$$

where the adversary $\mathcal{A}$ is said to be admissible if and only if:

1. $f^0(x_1^0, \ldots, x_n^0) = f^1(x_1^1, \ldots, x_n^1)$ *for all sequences* $(x_1^0, \ldots, x_n^0, x_1^1, \ldots, x_n^1, f^0, f^1, \mathsf{lab})$ *such that:*
   - *For all $i \in [n]$, $(i, \mathsf{lab}, x_i^0, x_i^1) \in \mathcal{MS}$,*
   - $(f^0, f^1) \in \mathcal{FS}$.
2. *When XX = pos, for any label* lab *queried by the adversary, $q_c[i, \mathsf{lab}] > 0$ for all $i \in [n]$, where $q_c[i, \mathsf{lab}]$ denotes the number of elements of the form $(i, \mathsf{lab}, *, *)$ in $\mathcal{MS}$.*
3. *When YY = mh, $f^{\nu,0} = f^{\nu,1}$ for all $\nu \in [q_k]$.*

Remark 2.3. In this paper, we only consider pos-security since a sel-pos-YY-secure MIFE/MCFE scheme can be generically transformed into a sel-any-YY-secure MIFE/MCFE scheme [1,2,5,23].

2.3 Functionalities

In this section, we define basic function classes for MIFE/SK-MCFE that is used in this paper.

Definition 2.4 (Inner Product over Bilinear Groups). *Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. A function family $\mathcal{F}^{\mathsf{IP}}_{m,n,\mathbb{G}}$ for inner products over bilinear groups consists of functions $f : (G_1^m)^n \to G_T$. Each $f \in \mathcal{F}^{\mathsf{IP}}_{m,n,\mathbb{G}}$ is specified by $[(\mathbf{y}_1, \ldots, \mathbf{y}_n)]_2$ where $\mathbf{y}_i \in \mathbb{Z}_p^m$ and defined as $f([\mathbf{x}_1]_1, \ldots, [\mathbf{x}_n]_1) = [\sum_{i \in [n]} \langle \mathbf{x}_i, \mathbf{y}_i \rangle]_T$. We call MIFE/SK-MCFE for $\mathcal{F}^{\mathsf{IP}}_{m,n,\mathbb{G}}$ MIFE/SK-MCFE for inner product. Especially, we sometimes call FE for $\mathcal{F}^{\mathsf{IP}}_{m,1,\mathbb{G}}$ inner product functional encryption (IPFE).*

Note that constructions of IPFE and SK-MCFE for inner product with function-hiding (sel-any-fh) security are already known [10,33].

Definition 2.5 (Mixed-Group Inner Products). *Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups. A function family $\mathcal{F}^{\mathsf{MG}}_{m_1,m_2,n,\mathbb{G}}$ for mixed-group inner products consists of functions $f : (G_1^{m_1} \times G_2^{m_2})^n \to G_T$. Each $f \in \mathcal{F}^{\mathsf{MG}}_{m_1,m_2,n,\mathbb{G}}$ is specified by $([\mathbf{y}_{1,1}]_2, [\mathbf{y}_{1,2}]_1, \ldots, [\mathbf{y}_{n,1}]_2, [\mathbf{y}_{n,2}]_1)$ where $\mathbf{y}_{i,1} \in \mathbb{Z}_p^{m_1}$ and $\mathbf{y}_{i,2} \in \mathbb{Z}_p^{m_2}$ and defined as $f(([\mathbf{x}_{1,1}]_1, [\mathbf{x}_{1,2}]_2), \ldots, ([\mathbf{x}_{n,1}]_1, [\mathbf{x}_{n,2}]_2)) = [\langle \mathbf{x}, \mathbf{y} \rangle]_T$ where $\mathbf{x} = (\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \ldots, \mathbf{x}_{n,1}, \mathbf{x}_{n,2})$ and $\mathbf{y} = (\mathbf{y}_{1,1}, \mathbf{y}_{1,2}, \ldots, \mathbf{y}_{n,1}, \mathbf{y}_{n,2})$. We call MIFE/SK-MCFE for $\mathcal{F}^{\mathsf{MG}}_{m_1,m_2,n,\mathbb{G}}$ MIFE/SK-MCFE for mixed-group inner product.*

Definition 2.6 (Bounded-Norm Quadratic functions over $\mathbb{Z}$). *A function family $\mathcal{F}^{\mathsf{QF}}_{m,n,X,C}$ for bounded-norm multi-input quadratic functions consist of functions $f : (\mathcal{X}^m)^n \to \mathbb{Z}$ where $\mathcal{X} = \{i \in \mathbb{Z} \mid |i| \leq X\}$. Each $f \in \mathcal{F}^{\mathsf{QF}}_{m,n,X,C}$ is specified by $\mathbf{c} \in \mathbb{Z}^{(mn)^2}$ s.t. $||\mathbf{c}||_\infty \leq C$ and $\mathbf{c}[(i,j,k,\ell)] = 0$ if $i \geq k$. Then, f specified by $\mathbf{c}$ is defined as $f(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \sum_{i,k \in [n], j,\ell \in [m]} \mathbf{c}[(i,j,k,\ell)] \mathbf{x}_i[j] \mathbf{x}_k[\ell]$. We call MIFE/SK-MCFE for $\mathcal{F}^{\mathsf{QF}}_{m,n,X,C}$ MIFE/SK-MCFE for quadratic functions.*

Remark 2.7. The original definition of quadratic functions in [9] provides that $\mathbf{c}$ is a vector s.t. $\mathbf{c}[(i,j,k,\ell)] = 0$ if $(i,j) > (k,\ell)$ instead of $i \geq k$. Actually, the functionality in Definition 2.6 implies the original functionality by defining $g(\mathbf{x}_1, \ldots, \mathbf{x}_n) = f(\mathbf{x}'_1, \ldots, \mathbf{x}'_n)$ where $\mathbf{x}'_i = (\mathbf{x}_i \otimes \mathbf{x}_i, \mathbf{x}_i, 1)$ and $f \in \mathcal{F}^{\mathsf{QF}}_{m,n,X,C}$.

Formally, our contribution in this paper is the constructions of MIFE and SK-MCFE for quadratic functions from pairings. Note that only an SK-MIFE scheme for quadratic functions based on pairings [9] is know prior to our work.

3 SK-MCFE for Mixed-Group Inner Product

In this section, we provide our construction for function-hiding SK-MCFE for mixed-group inner-product (Definition 2.5), which is used as a building block of our MIFE and SK-MCFE schemes for quadratic functions. The construction is similar to the function-hiding SK-MIFE for mixed-group inner-product in [9] by Agrawal, Goyal, and Tomida (AGT). Recall that the AGT SK-MIFE for mixed-group inner-product is obtained by combining a function-hiding SK-MIFE for inner-product and a function-hiding SK-FE for inner product. Our SK-MCFE for mixed-group inner-product is obtained by replacing a function-hiding SK-MIFE for inner-product in the AGT scheme with a function-hiding SK-MCFE for inner-product. Note that a function-hiding SK-MCFE for inner product can be obtained from a function-hiding MCFE scheme for inner product in [10] since SK-MCFE is the special case of MCFE. Additionally, while the MCFE scheme in [10] uses a hash function modeled as a random oracle in encryption, we can replace it with a PRF in the secret-key setting. The function-hiding SK-MCFE scheme for inner product without a random oracle is presented in Fig. 2.

Formally, we construct a function-hiding SK-MCFE scheme for $\mathcal{F}^{\mathsf{MG}}_{m_1,m_2,n,\mathbb{G}}$ with label space $\mathcal{L}$ from a function-hiding SK-MCFE scheme for $\mathcal{F}^{\mathsf{IP}}_{m,n,\mathbb{G}}$ with the same label space $\mathcal{L}$ and a function-hiding FE scheme for $\mathcal{F}^{\mathsf{IP}}_{m,1,\mathbb{G}}$ in a generic way.

Let $\mathsf{icFE} = (\mathsf{icSetup}, \mathsf{icEnc}, \mathsf{icKeyGen}, \mathsf{icDec})$ be a function-hiding SK-MCFE for $\mathcal{F}^{\mathsf{IP}}_{m,n,\mathbb{G}}$, and $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme (SK-FE for $\mathcal{F}^{\mathsf{IP}}_{m,1,\mathbb{G}}$). Then, our function-hiding SK-MCFE for mixed-group inner product $\mathcal{F}^{\mathsf{MG}}_{m_1,m_2,n,\mathbb{G}}$ is constructed as shown in Fig. 1.

$\underline{\mathsf{Setup}(1^\lambda, 1^n)}$: It generates master secret keys of icFE and iFE as follows:
$\mathsf{icPP}, \mathsf{icMSK} \leftarrow \mathsf{icSetup}(1^\lambda, 1^n), \quad (\mathsf{iPP}_1, \mathsf{iMSK}_1), \ldots, (\mathsf{iPP}_n, \mathsf{iMSK}_n) \leftarrow \mathsf{iSetup}(1^\lambda)$
where the vector lengths of icFE and iFE are set as $m_1 + m_2 + k + 1$ and $m_2 + k + 1$, respectively.
Note that $k \geq 2$ is the parameter of the bilateral MDDH assumption.
Then it outputs $\mathsf{PP}, \mathsf{MSK}$ as follows:
$\mathsf{PP} = (\mathsf{icPP}, \mathsf{iPP}_1, \ldots, \mathsf{iPP}_n),\ \mathsf{MSK} = (\mathsf{icMSK}, \mathsf{iMSK}_1, \ldots, \mathsf{iMSK}_n)$.

$\underline{\mathsf{Enc}(\mathsf{MSK}, i, \mathsf{lab}, ([\mathbf{x}_{i,1}]_1, [\mathbf{x}_{i,2}]_2))}$: It output CT_i as follows:
$\mathbf{z} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{x}}_{i,1} = (\mathbf{x}_{i,1}, 0^{m_2}, \mathbf{z}, 0) \in \mathbb{Z}_p^{m_1+m_2+k+1},\ \widetilde{\mathbf{x}}_{i,2} = (\mathbf{x}_{i,2}, -\mathbf{z}, 0) \in \mathbb{Z}_p^{m_2+k+1}$
$\mathsf{icCT}_i \leftarrow \mathsf{icEnc}(\mathsf{icMSK}, i, \mathsf{lab}, [\widetilde{\mathbf{x}}_{i,1}]_1),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, [\widetilde{\mathbf{x}}_{i,2}]_2),\ \mathsf{CT}_i = (\mathsf{icCT}_i, \mathsf{iSK}_i)$

$\underline{\mathsf{KeyGen}(\mathsf{MSK}, \{[\mathbf{y}_{i,1}]_2, [\mathbf{y}_{i,2}]_1\}_{i\in[n]})}$: It output SK as follows:
$\mathbf{a} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{y}}_{i,1} = (\mathbf{y}_{i,1}, 0^{m_2}, \mathbf{a}, 0) \in \mathbb{Z}_p^{m_1+m_2+k+1},\ \widetilde{\mathbf{y}}_{i,2} = (\mathbf{y}_{i,2}, \mathbf{a}, 0) \in \mathbb{Z}_p^{m_2+k+1},\ \widetilde{\mathbf{y}} = (\widetilde{\mathbf{y}}_{1,1}, \ldots, \widetilde{\mathbf{y}}_{n,1})$
$\mathsf{icSK} \leftarrow \mathsf{icKeyGen}(\mathsf{icMSK}, [\widetilde{\mathbf{y}}]_2),\ \mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_i, [\widetilde{\mathbf{y}}_{i,2}]_1),\ \mathsf{SK} = (\mathsf{icSK}, \{\mathsf{iCT}_i\}_{i\in[n]})$

$\underline{\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})}$: It output z as follows:
Outputs $\mathsf{icDec}(\mathsf{icCT}_1, \ldots, \mathsf{icCT}_n, \mathsf{icSK}) \prod_{i\in[n]} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_i)$

Fig. 1. Our mixed-group IP-MIFE scheme.

Due to the limit of the space, we present the correctness and the security proof in the full version.

4 SK-MCFE for Quadratic Functions

As explained in the technical overview, i) our MIFE scheme for quadratic functions can be generically obtained from the modified AGT SK-MIFE scheme for quadratic functions, which does not use a SK-FE scheme for *predicate* inner product; ii) the modified SK-MIFE scheme can be seen as the special case of our SK-MCFE scheme, where the label space consists of one element. Considering the above two facts, we first present our SK-MCFE scheme for quadratic functions to save the effort of presenting the security proof of the modified SK-MIFE scheme in the construction of our MIFE scheme.

4.1 Construction

Let $\mathsf{mgFE} = (\mathsf{mgSetup}, \mathsf{mgEnc}, \mathsf{mgKeyGen}, \mathsf{mgDec})$ be an SK-MCFE scheme for mixed-group inner product (Sect. 3) with label space $\mathcal{L}$, and $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme. Also, let $\mathsf{PRF} = \{\mathsf{PRF}_\lambda\}_{\lambda\in\mathbb{N}}$ be a PRF family where $\mathsf{PRF}_\lambda : \{0,1\}^\lambda \times \mathcal{L} \to \{0,1\}^\lambda$ and $\mathbb{G}$ be bilinear groups. Below we provide an SK-MCFE scheme for function class $\mathcal{F}^{\mathsf{QF}}_{m,n,X,C}$

with the same label space $\mathcal{L}$. Similarly to [9], we can construct our SK-MCFE scheme from MDDH_k, while it makes the construction and security proof far more complicated as we can see in [9]. Thus, we present the construction based on MDDH_1 for better readability in this paper.

$\mathsf{Setup}(1^\lambda, 1^n)$ samples a random PRF key $K \leftarrow \{0,1\}^\lambda$ and the master keys for the underlying IPFE and SK-MCFE scheme as $(\mathsf{iPP}^{(2)}, \mathsf{iMSK}^{(2)}) \leftarrow \mathsf{iSetup}(1^\lambda)$, $(\mathsf{mgPP}, \mathsf{mgMSK}) \leftarrow \mathsf{mgSetup}(1^\lambda, 1^n)$ where the vector length of iFE is set as 2, and the vector length of mgFE is set as m^2n+2 and 1. Note that $\mathsf{iPP}^{(2)} = \mathsf{mgPP} = \mathbb{G}$. It also samples a sequence of randomization terms as:

$$\begin{aligned} &\forall\ i,k \in [n], j,\ell \in [m], && w_{(i,j,k,\ell)} \leftarrow \mathbb{Z}_p \\ &\forall\ i \in [n], j \in [m], && u_{i,j}, \widetilde{u}_{i,j}, v_{i,j}, \widetilde{v}_{i,j} \leftarrow \mathbb{Z}_p \end{aligned}$$

It outputs the public parameters and master key as

$$\mathsf{PP} = \mathbb{G}, \ \ \mathsf{MSK} = \left(K, \mathsf{iMSK}^{(2)}, \mathsf{mgMSK}, \{w_{(i,j,k,\ell)}\}_{i,j,k,\ell}, \{u_{i,j}, \widetilde{u}_{i,j}, v_{i,j}, \widetilde{v}_{i,j}\}_{i,j}\right).$$

$\mathsf{Enc}(\mathsf{MSK}, i, \mathsf{lab}, \mathbf{x})$ parses MSK as above, and using the PRF key K, it samples a IPFE master key of vector length $mn+3m+4$ as $(\mathsf{iPP}^{(1)}, \mathsf{iMSK}^{(1)}) \leftarrow \mathsf{iSetup}(1^\lambda; \mathsf{PRF}(K, \mathsf{lab}))$. Here we assume (w.l.o.g.) that the MIFE setup algorithm takes λ bits as random coins. It then samples random elements $s, \widetilde{s}, r, t \leftarrow \mathbb{Z}_p$. And, it sets vectors $\mathbf{b}_j, \widetilde{\mathbf{b}}_j$ for $j \in [m]$ as follows:

$$\mathbf{b}_j = (\mathbf{x}[j], 0, s\mathbf{e}_{(i,j)}, ru_{i,j}, v_{i,j}, \mathbf{0}_{3m}), \quad \widetilde{\mathbf{b}}_j = (\mathbf{x}[j], 0, \widetilde{s}\mathbf{w}_{(*,*,i,j)}, \widetilde{u}_{i,j}, t\widetilde{v}_{i,j}, \mathbf{0}_{3m}).$$

where $\mathbf{e}_{(i,j)}$ is the mn-dimensional one-hot vector with the (i,j)-th element being 1, and vector $\mathbf{w}_{(*,*,i,j)} \in \mathbb{Z}_p^{mn}$ is defined as follows:

$$\forall\ j \in [m], \quad \mathbf{w}_{(*,*,i,j)} = \left(w_{(1,1,i,j)}, w_{(1,2,i,j)}, \ldots, w_{(n,m,i,j)}\right)$$

The encryptor encodes the vectors $\mathbf{b}_j, \widetilde{\mathbf{b}}_j$ under MIFE as follows:

$$\forall\ j \in [m], \ \ \mathsf{iCT}_j \leftarrow \mathsf{iEnc}(\mathsf{iMSK}^{(1)}, [\mathbf{b}_j]_1), \ \ \mathsf{iSK}_j \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}^{(1)}, [\widetilde{\mathbf{b}}_j]_2).$$

It also encodes the random elements $s, \widetilde{s}$ as follows:

$$\mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}^{(2)}, [(s,0)]_1), \ \ \mathsf{iSK} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}^{(2)}, [(\widetilde{s},0)]_2).$$

Lastly, it sets $\mathbf{f} = (r, t, \mathbf{0}_{m^2n})$, $h = 0$, and encrypts elements $\mathbf{f}, h$ as

$$\mathsf{mgCT} \leftarrow \mathsf{mgEnc}(\mathsf{mgMSK}, i, \mathsf{lab}, ([\mathbf{f}]_1, [h]_2)).$$

And the resulting ciphertext is set as below:

$$\mathsf{CT} = \left(\{\mathsf{iCT}_j\}_j, \{\mathsf{iSK}_j\}_j, \mathsf{iCT}, \mathsf{iSK}, \mathsf{mgCT}\right).$$

$\mathsf{KeyGen}(\mathsf{MSK}, \mathbf{c})$ parses MSK as above, and the key vector $\mathbf{c}$ lies in the space $\mathbb{Z}^{(mn)^2}$. Let the vector $\widetilde{\mathbf{f}}_i \in \mathbb{Z}_p^{(2+m^2n)}$ be the following vector: for all $i \in [n]$,

$$\widetilde{\mathbf{f}}_i[1] = \sum_{j,\ell\in[m],k\in[n]} \mathbf{c}[(i,j,k,\ell)]u_{i,j}\widetilde{u}_{k,\ell},\ \widetilde{\mathbf{f}}_i[2] = \sum_{j,\ell\in[m],k\in[n]} \mathbf{c}[(k,\ell,i,j)]v_{k,\ell}\widetilde{v}_{i,j}$$

and $\widetilde{\mathbf{f}}_i$ is zeros at all other places. It also sets $\widetilde{h}_i = 0$ for all $i \in [n]$. The key generator samples a SK-MCFE secret key corresponding to vectors $\{\widetilde{\mathbf{f}}_i, \widetilde{h}_i\}_i$ as $\mathsf{mgSK} \leftarrow \mathsf{mgKeyGen}(\mathsf{mgMSK}, \{[\widetilde{\mathbf{f}}_i]_2, [\widetilde{h}_i]_1\}_{i\in[n]})$, and partial derandomization terms:

$$\forall\ i,k \in [n], \quad \sigma_{i,k} = \sum_{j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)]w_{(i,j,k,\ell)}$$

And, it outputs the secret key as

$$\mathsf{SK} = (\mathbf{c}, \mathsf{mgSK}, \{\sigma_{i,k}\}_{i,k}).$$

$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})$ parses the ciphertexts and secret key as:

$$\mathsf{CT}_i = (\{\mathsf{iCT}_{i,j}\}_{i,j}, \{\mathsf{iSK}_{i,j}\}_{i,j}, \mathsf{iCT}_i, \mathsf{iSK}_i, \mathsf{mgCT}_i),\ \ \mathsf{SK} = (\mathbf{c}, \mathsf{mgSK}, \{\sigma_{i,k}\}_{i,k}).$$

It runs the MIFE decryption algorithm as:

$$[z_1]_T = \prod_{\substack{i,k\in[n]\\ j,\ell\in[m]}} \mathsf{iDec}(\mathsf{iCT}_{i,j}, \mathsf{iSK}_{k,\ell})^{\mathbf{c}[(i,j,k,\ell)]},\ [z_2]_T = \prod_{i,k\in[n]} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_k)^{\sigma_{i,k}}$$

It also runs the SK-MCFE decryption algorithm as:

$$[z_3]_T = \mathsf{mgDec}(\mathsf{mgCT}_1, \ldots, \mathsf{mgCT}_n, \mathsf{mgSK})$$

Finally it outputs z where $[z]_T = [z_1 - z_2 - z_3]_T$ by searching for z within the range of $z \le |m^2n^2CX^2|$.

Correctness. Let $s_i, \widetilde{s}_i, r_i, t_i$ for $i \in [n]$ be random elements used to generate CT_i. Due to the correctness of $\mathsf{iFE}, \mathsf{mgFE}$, in decryption, we have

$$z_1 = \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)](\mathbf{x}_i[j]\mathbf{x}_k[\ell] + s_i\widetilde{s}_k w_{(i,j,k,\ell)} + r_i u_{i,j}\widetilde{u}_{k,\ell} + t_k v_{i,j}\widetilde{v}_{k,\ell})$$

$$z_2 = \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)]s_i\widetilde{s}_k w_{(i,j,k,\ell)}$$

$$z_3 = \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)](r_i u_{i,j}\widetilde{u}_{k,\ell} + t_k v_{i,j}\widetilde{v}_{k,\ell}).$$

Therefore, we have $z = \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)]\mathbf{x}_i[j]\mathbf{x}_k[\ell]$.

4.2 Security

For security, we have the following theorem.

Theorem 4.1. *If* iFE *and* mgFE *are sel-pos-fh-IND-secure, and the MDDH*$_1$ *assumption holds in* $\mathbb{G}$*, then the proposed SK-MCFE for quadratic functions is sel-pos-mh-IND-secure.*

Due to the limit of the space, we present the security proof in the full version.

5 MIFE for Quadratic Functions

In this section, we provide our construction for MIFE for quadratic functions.

5.1 Homomorphism in Underlying Schemes

For the construction of our MIFE for quadratic functions, we use the same building blocks as SK-MCFE for quadratic functions Sect. 4, namely, a function-hiding SK-MCFE scheme mgFE for mixed-group inner product and a function-hiding IPFE scheme iFE. Additionally, we require them to have homomorphism for the construction of MIFE for quadratic functions. Precisely iFE needs to have homomorphism for both encryption and key generation while mgFE needs to have homomorphism for only encryption.

Homomorphism of iFE. We use function-hiding IPFE in [33] for iFE with homomorphism. In their construction from MDDH$_1$, the setup algorithm chooses a bilinear group $\mathbb{G}$ and a random matrix $\mathbf{B}$ in $\mathbb{Z}_p^{(m+3)\times(m+3)}$, and sets $\mathsf{PP} = \mathbb{G}, \mathsf{MSK} = (\mathbf{B}, \mathbf{B}^*)$ where $\mathbf{B}^* = (\mathbf{B}^{-1})^\top$. Encryption of $[\mathbf{x}]_1 \in G_1^m$ chooses $r \leftarrow \mathbb{Z}_p$ and outputs $\mathsf{iCT} = [(\mathbf{x}, r, 0, 0)\mathbf{B}]_1$. Similarly, key generation of $[\mathbf{y}]_2 \in G_2^m$ chooses $s \leftarrow \mathbb{Z}_p$ and outputs $\mathsf{iSK} = [(\mathbf{y}, 0, s, 0)\mathbf{B}^*]_2$. Thus, the random-tape space of iEnc and $\mathsf{iKeyGen}$ can be seen as $\mathbb{Z}_p$ and, for all $\mathbf{x}_1, \mathbf{x}_2, \mathbf{y}_1, \mathbf{y}_2 \in \mathbb{Z}_p^m, a_1, a_2, r_1, r_2, s_1, s_2 \in \mathbb{Z}_p$ we have the following homomorphism of $\mathbb{Z}_p$-module with respect to encryption and key generation:

$$\begin{aligned} a_1\mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{x}_1]_1; r_1) + a_2\mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{x}_2]_1; r_2) \\ = \mathsf{iEnc}(\mathsf{iMSK}, [a_1\mathbf{x}_1 + a_2\mathbf{x}_2]_1; a_1r_1 + a_2r_2) \\ a_1\mathsf{iKeyGen}(\mathsf{iMSK}, [\mathbf{y}_1]_2; s_1) + a_2\mathsf{iKeyGen}(\mathsf{iMSK}, [\mathbf{y}_2]_2; s_2) \\ = \mathsf{iKeyGen}(\mathsf{iMSK}, [a_1\mathbf{y}_1 + a_2\mathbf{y}_2]_2; a_1s_1 + a_2s_2) \end{aligned}$$

We can confirm this as follows:

$$a_1[(\mathbf{x}_1, r_1, 0, 0)\mathbf{B}]_1 + a_2[(\mathbf{x}_2, r_2, 0, 0)\mathbf{B}]_1 = [(a_1\mathbf{x}_1 + a_2\mathbf{x}_2, a_1r_1 + a_2r_2, 0, 0)\mathbf{B}]_1$$

$$a_1[(\mathbf{y}_1, 0, s_1, 0)\mathbf{B}^*]_2 + a_2[(\mathbf{y}_2, 0, s_2, 0)\mathbf{B}^*]_2 = [(a_1\mathbf{y}_1 + a_2\mathbf{y}_2, 0, a_1s_1 + a_2s_2, 0)\mathbf{B}^*]_2.$$

Homomorphism of mgFE. As shown in Sect. 3, our SK-MCFE scheme mgFE for mixed-group inner product uses a function-hiding SK-MCFE scheme icFE for inner product and function-hiding FE scheme iFE for inner product as a building block. For a function-hiding SK-MCFE scheme for inner product, we use a slightly modified function-hiding MCFE scheme for inner product proposed in [10], which is described in Fig. 2. The modification lies in the way of generating t in encryption, which is generated via a random oracle in the MCFE scheme in [10], but PRF suffices in the secret-key setting. Since an icFE ciphertext consists of a iFE ciphertext, a mgFE ciphertext of $([\mathbf{x}_1]_1, [\mathbf{x}_2]_2) \in G_1^{m_1} \times G_2^{m_2}$ can be generated as

$$\begin{aligned}
&r_1, r_2 \leftarrow \mathbb{Z}_p, \ \ \mathbf{z} \leftarrow \mathbb{Z}_p^k, \ \ t = \mathsf{PRF}(K, \mathsf{lab}) \\
&\mathsf{iEnc}(\mathsf{iMSK}^{(1)}, ([(\mathbf{x}_1, 0^{m_2}, \mathbf{z}, 0, 0^{m_1+m_2+k+1}, t, 0)]_1); r_1) \\
&\mathsf{iKeyGen}(\mathsf{iMSK}^{(2)}, ([(\mathbf{x}_2, -\mathbf{z}, 0,)]_2); r_2)
\end{aligned}$$

for some master secret keys $\mathsf{iMSK}^{(1)}, \mathsf{iMSK}^{(2)}$ and PRF key K. Thus, the random-tape space of mgEnc can be set as $\mathbb{Z}_p^{k+2}$, and by using the homomorphism of iFE, we can obtain the following homomorphism of ciphertexts in mgFE. For all $N \in \mathbb{N}$, $i \in [n]$, $\mathsf{lab} \in \mathcal{L}$, $a_1, \ldots, a_N \in \mathbb{Z}_p$ s.t. $\sum_{j \in [N]} a_j = 1$, $\mathbf{x}_{1,1}, \ldots, \mathbf{x}_{N,1} \in \mathbb{Z}_p^{m_1}$, $\mathbf{x}_{1,2}, \ldots, \mathbf{x}_{N,2} \in \mathbb{Z}_p^{m_2}$, $\mathbf{r}_1, \ldots, \mathbf{r}_N \in \mathbb{Z}_p^{k+2}$, we have

$$\begin{aligned}
&\sum_{j \in [N]} a_j \mathsf{mgEnc}(\mathsf{mgMSK}, i, \mathsf{lab}, ([\mathbf{x}_{j,1}]_1, [\mathbf{x}_{j,2}]_2); \mathbf{r}_j) \\
= \ &\mathsf{mgEnc}(\mathsf{mgMSK}, i, \mathsf{lab}, ([\sum_{j \in [N]} a_j \mathbf{x}_{j,1}]_1, [\sum_{j \in [N]} a_j \mathbf{x}_{j,2}]_2); \sum_{j \in [N]} a_j \mathbf{r}_j)
\end{aligned}$$

5.2 Construction

Let $\mathsf{mgFE} = (\mathsf{mgSetup}, \mathsf{mgEnc}, \mathsf{mgKeyGen}, \mathsf{mgDec})$ be an SK-MCFE scheme for mixed-group inner product (Sect. 3) with label space $\mathcal{L}$, and $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme. Also, let $\mathsf{PRF} = \{\mathsf{PRF}_\lambda\}_{\lambda \in \mathbb{N}}$ be a PRF family where $\mathsf{PRF}_\lambda : \{0,1\}^\lambda \times \mathcal{L} \to \{0,1\}^\lambda$. Let lab_0 be a fixed label in $\mathcal{L}$ and $D = 4m + 2k + 17$ where k is the parameter for the bilateral MDDH assumption used for mgFE. Below we provide an MIFE scheme for function class $\mathcal{F}^{\mathsf{QF}}_{m,n,X,C}$. Note that MstEnc is a subroutine algorithm used in Setup, which corresponds to $\widetilde{\mathsf{Enc}}$ of property P in the technical overview.

$\mathsf{Setup}(1^\lambda, 1^n)$: Let $\mathsf{PRF} = \{\mathsf{PRF}_\lambda\}_{\lambda\in\mathbb{N}}$ be a PRF family where $\mathsf{PRF}_\lambda : \{0,1\}^\lambda \times \mathcal{L} \to \mathbb{Z}_p$. On input the security parameter 1^λ and the number of slots 1^n, the setup algorithm outputs $(\mathsf{PK}, \mathsf{MSK})$ as follows.

$$\{\mathsf{iPP}_i, \mathsf{iMSK}_i \leftarrow \mathsf{iSetup}(1^\lambda)\}_{i\in[n]}, \quad K \leftarrow \{0,1\}^\kappa$$
$$\mathsf{PP} = \{\mathsf{PP}_i\}_{i\in[n]},\ \mathsf{MSK} = (K, \{\mathsf{MSK}_i\}_{i\in[n]}).$$

$\mathsf{Enc}(\mathsf{MSK}, i, \mathsf{lab}, [\mathbf{x}_i]_1)$: The encryption algorithm takes as input user MSK, user index $i \in [n]$, an input vector $[\mathbf{x}_i]_1$, a label lab and outputs CT_i as follows.

$$t = \mathsf{PRF}(K, \mathsf{lab}),\ \widehat{\mathbf{x}}_i = (\mathbf{x}_i, 0^m, t, 0),\ \mathsf{CT}_i = \mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}_i, [\widehat{\mathbf{x}}_i]_1).$$

$\mathsf{KeyGen}(\mathsf{MSK}, \{[\mathbf{y}_i]_2\}_{i\in[n]})$: The key generation algorithm takes as input the master secret key MSK, and vectors $\{[\mathbf{y}_i]_2\}_{i\in[n]}$ and outputs SK as follows. It randomly chooses $r_i \in \mathbb{Z}_p$ so that $\sum_{i\in[n]} r_i = 0$ and compute

$$\widehat{\mathbf{y}}_i = (\mathbf{y}_i, 0^m, r_i, 0),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}_i, [\widehat{\mathbf{y}}_i]_2),\ \mathsf{SK} = \{\mathsf{iSK}_i\}_{i\in[n]}.$$

$\mathsf{Dec}(\mathsf{SK}, \mathsf{CT}_1, ..., \mathsf{CT}_n)$: The decryption algorithm takes as input the secret key SK, ciphertexts $\mathsf{CT}_1, \ldots, \mathsf{CT}_n$ and outputs d as follows.

$$[d]_T = \prod_{i\in[n]} \mathsf{iDec}(\mathsf{iSK}_i, \mathsf{iCT}_i).$$

Fig. 2. Function-Hiding SK-MCFE for inner product

$\mathsf{Setup}(1^\lambda, 1^n)$ samples a random PRF key $K \leftarrow \{0,1\}^\lambda$ and the master keys for the underlying IPFE and SK-MCFE scheme as $(\mathsf{iPP}^{(2)}, \mathsf{iMSK}^{(2)}) \leftarrow \mathsf{iSetup}(1^\lambda)$, $(\mathsf{mgPP}, \mathsf{mgMSK}) \leftarrow \mathsf{mgSetup}(1^\lambda, 1^n)$ where the vector length of iFE is set as 2, and the vector length of mgFE is set as $m^2n + 2$ and 1. Note that $\mathsf{iPP}^{(2)} = \mathsf{mgPP} = \mathbb{G}$. It also samples a sequence of randomization terms as:

$$\forall\ i, k \in [n], j, \ell \in [m], \qquad w_{(i,j,k,\ell)} \leftarrow \mathbb{Z}_p$$
$$\forall\ i \in [n], j \in [m], \qquad u_{i,j}, \widetilde{u}_{i,j}, v_{i,j}, \widetilde{v}_{i,j} \leftarrow \mathbb{Z}_p$$

It sets the public parameters and master key as

$$\mathsf{PP} = \mathbb{G},\ \ \mathsf{MSK} = \left(K, \mathsf{iMSK}^{(2)}, \mathsf{mgMSK}, \{w_{(i,j,k,\ell)}\}_{i,j,k,\ell}, \{u_{i,j}, \widetilde{u}_{i,j}, v_{i,j}, \widetilde{v}_{i,j}\}_{i,j}\right).$$

It runs MstEnc described below to generate master ciphertexts, which forms encryption keys, as

$$\forall\ i \in [n], j \in [m], \quad \mathsf{MCT}_{1,i,j} \leftarrow \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{e}_j)$$
$$\forall\ i \in [n], j \in [D], \quad \mathsf{MCT}_{0,i,j} \leftarrow \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{0}_m).$$

Finally it output encryption keys together with the public key and master secret key as

$$\forall\ i \in [n], \quad \mathsf{EK}_i = (\{\mathsf{MCT}_{1,i,j}\}_{j\in[n]}, \{\mathsf{MCT}_{0,i,j}\}_{j\in[D]}).$$

$\mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{x})$ parses MSK as above, and using the PRF key K, it samples a IPFE master key of vector length $mn + 3m + 4$ as:

$$(\mathsf{iPP}^{(1)}, \mathsf{iMSK}^{(1)}) \leftarrow \mathsf{iSetup}(1^\lambda; \mathsf{PRF}(K, \mathsf{lab}_0))$$

It then samples random elements $s, \widetilde{s}, r, t \leftarrow \mathbb{Z}_p$. And, it sets vectors $\mathbf{b}_j, \widetilde{\mathbf{b}}_j$ for $j \in [m]$ as follows:

$$\mathbf{b}_j = (\mathbf{x}[j], 0, s\mathbf{e}_{(i,j)}, ru_{i,j}, v_{i,j}, \mathbf{0}_{3m}), \quad \widetilde{\mathbf{b}}_j = (\mathbf{x}[j], 0, \widetilde{s}\mathbf{w}_{(*,*,i,j)}, \widetilde{u}_{i,j}, t\widetilde{v}_{i,j}, \mathbf{0}_{3m}).$$

where where $\mathbf{e}_{(i,j)}$ is the mn-dimensional one-hot vector with the (i, j)-th element being 1, and vector $\mathbf{w}_{(*,*,i,j)} \in \mathbb{Z}_p^{mn}$ is defined as follows:

$$\forall\ j \in [m], \quad \mathbf{w}_{(*,*,i,j)} = \left(w_{(1,1,i,j)}, w_{(1,2,i,j)}, \ldots, w_{(n,m,i,j)}\right)$$

The encryptor encodes the vectors $\mathbf{b}_j, \widetilde{\mathbf{b}}_j$ under MIFE as follows:

$$\forall\ j \in [m], \quad \mathsf{iCT}_j \leftarrow \mathsf{iEnc}(\mathsf{iMSK}^{(1)}, [\mathbf{b}_j]_1), \ \ \mathsf{iSK}_j \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}^{(1)}, [\widetilde{\mathbf{b}}_j]_2).$$

It also encodes the random elements $s, \widetilde{s}$ as follows:

$$\mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}^{(2)}, [(s, 0)]_1), \ \ \mathsf{iSK} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}^{(2)}, [(\widetilde{s}, 0)]_2).$$

Lastly, it sets $\mathbf{f} = (r, t, \mathbf{0}_{m^2n}),\ h = 0$, and encrypts elements $\mathbf{f}, h$ with respect to label lab_0 as

$$\mathsf{mgCT} \leftarrow \mathsf{mgEnc}(\mathsf{mgMSK}, i, \mathsf{lab}_0, ([\mathbf{f}]_1, [h]_2)).$$

The resulting ciphertext is set as $\mathsf{MCT} = (\{\mathsf{iCT}_j\}_j, \{\mathsf{iSK}_j\}_j, \mathsf{iCT}, \mathsf{iSK}, \mathsf{mgCT})$.

$\mathsf{Enc}(\mathsf{EK}_i, \mathbf{x})$ parses EK_i as above. It then samples random elements $\gamma_1, \ldots, \gamma_{(D-1)/2} \leftarrow \mathbb{Z}_p$. And, it encrypts $\mathbf{x}$ to CT by homomorphic addition of master ciphertexts as follows:

$$\mathsf{CT} = \sum_{j\in[m]} \mathbf{x}[j]\mathsf{MCT}_{1,i,j} - \left(\sum_{j\in[m]} \mathbf{x}[j] - 1\right)\mathsf{MCT}_{0,i,1} + \sum_{j\in[(D-1)/2]} \gamma_j(\mathsf{MCT}_{0,i,2j} - \mathsf{MCT}_{0,i,2j+1})$$

where the above is the component-wise homomorphic addition with respect to ciphertexts of iFE and mgFE. Then, it outputs CT.

$\mathsf{KeyGen}(\mathsf{MSK}, \mathbf{c})$ parses MSK as above, and the key vector $\mathbf{c}$ lies in the space $\mathbb{Z}_p^{(mn)^2}$. Let the vector $\widetilde{\mathbf{f}}_i \in \mathbb{Z}_p^{(2+m^2n)}$ be the following vector: for all $i \in [n]$

$$\widetilde{\mathbf{f}}_i[1] = \sum_{j,\ell \in [m], k \in [n]} \mathbf{c}[(i,j,k,\ell)] u_{i,j} \widetilde{u}_{k,\ell},\ \widetilde{\mathbf{f}}_i[2] = \sum_{j,\ell \in [m], k \in [n]} \mathbf{c}[(k,\ell,i,j)] v_{k,\ell} \widetilde{v}_{i,j}$$

and $\widetilde{\mathbf{f}}_i$ is zeros at all other places. It also sets $\widetilde{h}_i = 0$ for all $i \in [n]$. The key generator samples a SK-MCFE secret key corresponding to vectors $\{\widetilde{\mathbf{f}}_i, \widetilde{h}_i\}_i$ as $\mathsf{mgSK} \leftarrow \mathsf{mgKeyGen}(\mathsf{mgMSK}, \{[\widetilde{\mathbf{f}}_i]_2, [\widetilde{h}_i]_1\}_{i \in [n]})$, and partial derandomization terms:

$$\forall\ i, k \in [n], \quad \sigma_{i,k} = \sum_{j,\ell \in [m]} \mathbf{c}[(i,j,k,\ell)] w_{(i,j,k,\ell)}$$

And, it outputs the secret key as $\mathsf{SK} = (\mathbf{c}, \mathsf{mgSK}, \{\sigma_{i,k}\}_{i,k})$.

$\mathsf{Dec}(\mathsf{CT}_1, \ldots, \mathsf{CT}_n, \mathsf{SK})$ parses the ciphertexts and secret key as:

$$\mathsf{CT}_i = (\{\mathsf{iCT}_{i,j}\}_{i,j}, \{\mathsf{iSK}_{i,j}\}_{i,j}, \mathsf{iCT}_i, \mathsf{iSK}_i, \mathsf{mgCT}_i),$$
$$\mathsf{SK} = (\mathbf{c}, \mathsf{mgSK}, \{\sigma_{i,k}\}_{i,k}).$$

It runs the MIFE decryption algorithm as:

$$[z_1]_T = \prod_{\substack{i,k \in [n], \\ j,\ell \in [m]}} \mathsf{iDec}(\mathsf{iCT}_{i,j}, \mathsf{iSK}_{k,\ell})^{\mathbf{c}[(i,j,k,\ell)]},\ [z_2]_T = \prod_{i,k \in [n]} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_k)^{\sigma_{i,k}}$$

It also runs the SK-MCFE decryption algorithm as:

$$[z_3]_T = \mathsf{mgDec}(\mathsf{mgCT}_1, \ldots, \mathsf{mgCT}_n, \mathsf{mgSK})$$

Finally it outputs z where $[z]_T = [z_1 - z_2 - z_3]_T$ by searching for z within the range of $z \leq |m^2n^2CX^2|$.

Correctness. Let $s_{b,i,j}, \widetilde{s}_{b,i,j}, r_{b,i,j}, t_{b,i,j}$ for $b \in \{0,1\}, i \in [n], j \in [D]$ be random elements used to generate $\mathsf{MCT}_{b,i,j}$ in EK_i. Thanks to the homomorphism of iFE and mgFE, $\mathsf{Enc}(\mathsf{EK}_i, \mathbf{x})$ outputs $\mathsf{CT}_i = (\{\mathsf{iCT}_{i,j}\}_j, \{\mathsf{iSK}_{i,j}\}_j, \mathsf{iCT}_i, \mathsf{iSK}_i, \mathsf{mgCT}_i)$, which are encryption of

$$\begin{aligned} &[\mathbf{b}]_1 = [(\mathbf{x}_i[j], 0, s_i \mathbf{e}_{(i,j)}, r_i u_{i,j}, v_{i,j}, \mathbf{0}_{3m})]_1 \\ &[\widetilde{\mathbf{b}}]_2 = [(\mathbf{x}_i[j], 0, \widetilde{s}_i \mathbf{w}_{(*,*,i,j)}, \widetilde{u}_{i,j}, t_i \widetilde{v}_{i,j}, \mathbf{0}_{3m})]_2 \\ &[(s_i, 0)]_1, \quad [(\widetilde{s}_i, 0)]_2, \quad ([\mathbf{f}]_1, [h]_2) = ([(r_i, t_i, \mathbf{0}_{m^2n})]_1, [0]_2) \text{ for label } \mathsf{lab}_0 \end{aligned} \tag{5}$$

respectively, where

$$
\begin{aligned}
s_i &= \sum_{j\in[m]} \mathbf{x}_i[j]s_{1,i,j} - \left(\sum_{j\in[m]} \mathbf{x}_i[j]-1\right) s_{0,i,1} + \sum_{j\in[(D-1)/2]} \gamma_j(s_{0,i,2j}-s_{0,i,2j+1})\\
\widetilde{s}_i &= \sum_{j\in[m]} \mathbf{x}_i[j]\widetilde{s}_{1,i,j} - \left(\sum_{j\in[m]} \mathbf{x}_i[j]-1\right) \widetilde{s}_{0,i,1} + \sum_{j\in[(D-1)/2]} \gamma_j(\widetilde{s}_{0,i,2j}-\widetilde{s}_{0,i,2j+1})\\
r_i &= \sum_{j\in[m]} \mathbf{x}_i[j]r_{1,i,j} - \left(\sum_{j\in[m]} \mathbf{x}_i[j]-1\right) r_{0,i,1} + \sum_{j\in[(D-1)/2]} \gamma_j(r_{0,i,2j}-r_{0,i,2j+1})\\
t_i &= \sum_{j\in[m]} \mathbf{x}_i[j]t_{1,i,j} - \left(\sum_{j\in[m]} \mathbf{x}_i[j]-1\right) t_{0,i,1} + \sum_{j\in[(D-1)/2]} \gamma_j(t_{0,i,2j}-t_{0,i,2j+1}).
\end{aligned}
\tag{6}
$$

Hence, similarly to the correctness of our SK-MCFE for quadratic functions (Sect. 4), in decryption, we have

$$
\begin{aligned}
z_1 &= \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)](\mathbf{x}_i[j]\mathbf{x}_k[\ell] + s_i\widetilde{s}_k w_{(i,j,k,\ell)} + r_i u_{i,j}\widetilde{u}_{k,\ell} + t_k v_{i,j}\widetilde{v}_{k,\ell})\\
z_2 &= \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)]s_i\widetilde{s}_k w_{(i,j,k,\ell)}\\
z_3 &= \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)](r_i u_{i,j}\widetilde{u}_{k,\ell} + t_k v_{i,j}\widetilde{v}_{k,\ell}).
\end{aligned}
$$

Since $\mathbf{c}[(i,j,k,\ell)] = 0$ for $i \geq k$, we have $z = \sum_{i,k\in[n],j,\ell\in[m]} \mathbf{c}[(i,j,k,\ell)]\, \mathbf{x}_i[j]\mathbf{x}_k[\ell]$.

5.3 Security

For security, we have the following theorem. Let qcFE be SK-MCFE scheme for quadratic functions in Sect. 4.

Theorem 5.1. *If* qcFE *are sel-pos-mh-IND-secure, then the proposed MIFE for quadratic functions is sel-pos-mh-IND-secure.*

Proof. Wlog, in the pos setting, we can denote challenge messages by $\{i, \mathbf{x}_i^{\mu,0}, \mathbf{x}_i^{\mu,1}\}_{i\in[n],\mu\in[q_c]}$ for some q_c instead of $\{i^\mu, \mathbf{x}_{i^\mu}^{\mu,0}, \mathbf{x}_{i^\mu}^{\mu,1}\}_{\mu\in[q_c']}$. For notational convenience, we use the former notation in this proof. We prove Theorem 5.1 via a series of hybrids $\mathsf{H}_1^\beta, \mathsf{H}_\mathsf{f}^\beta$. We show that $\mathsf{H}_0^\beta \approx_c \mathsf{H}_1^\beta \approx_c \mathsf{H}_\mathsf{f}^\beta$, where H_0^β is the original security game for MIFE defined in Definition 2.1. Each hybrid is defined as described in Fig. 3, where the reply for the ciphertext query is computed by MstEnc instead of Enc. We denote the probability that $\mathcal{A}$ outputs β in hybrid H^β by $\mathsf{P}(\mathcal{A}, \mathsf{H}^\beta)$ in what follows.

Theorem 5.1 directly follows from Lemma 5.2 and Lemma 5.3 since $\mathcal{A}$ does not obtain the information on β in $\mathsf{H}_\mathsf{f}^\beta$. □

```
$\mathsf{H}_0^\beta$, $\mathsf{H}_1^\beta$, $\mathsf{H}_\mathsf{f}^\beta$
$\beta \leftarrow \{0,1\},\ \mathsf{PP}, \{\mathsf{EK}_i\}_{i\in[n]}, \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda)$
$(\mathcal{CS}, \{i, \mathbf{x}_i^{\mu,0}, \mathbf{x}_i^{\mu,1}\}_{i\in[n],\mu\in[q_c]}, \{\mathbf{c}^\nu\}_{\nu\in[q_k]}) \leftarrow \mathcal{A}(\mathsf{PP})$
$\{\mathsf{CT}_i^\mu \leftarrow \mathsf{Enc}(\mathsf{EK}_i, \mathbf{x}_i^{\mu,\beta})\}_{i,\mu}$
 $\{\mathsf{CT}_i^\mu \leftarrow \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{x}_i^{\mu,\beta})\}_{i,\mu}$
 $\{\mathsf{CT}_i^\mu \leftarrow \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{x}_i^{\mu,0})\}_{i,\mu}$
$\{\mathsf{SK}^\nu \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, \mathbf{c}^\nu)\}_\nu$
$\beta' \leftarrow \mathcal{A}(\{\mathsf{EK}_i\}_{i\in\mathcal{CS}}, \{\mathsf{CT}_i^\mu\}_{i,\mu}, \{\mathsf{SK}^\nu\}_\nu)$
```

Fig. 3. Description of hybrids

Lemma 5.2. *For all PPT adversaries $\mathcal{A}$, we have $|\mathsf{P}(\mathcal{A}, \mathsf{H}_0^\beta) - \mathsf{P}(\mathcal{A}, \mathsf{H}_1^\beta)| \leq 2^{-\Omega(\lambda)}$.*

Proof. The difference between H_0^β and H_1^β lies in the way of generating challenge ciphertexts. That is, the challenge ciphertexts are generated by Enc in H_0^β while they are generated by MstEnc in H_1^β. Recall that the random elements used in MstEnc are $s, \widetilde{s}, r, t \in \mathbb{Z}_p$ and the random tapes used to generate $\big(\{\mathsf{iCT}_\ell\}_{\ell\in[m]}, \{\mathsf{iSK}_{\ell\in[m]}\}_\ell, \mathsf{iCT}, \mathsf{iSK}, \mathsf{mgCT}\big)$. Since iEnc, $\mathsf{iKeyGen}$ can use a random element in $\mathbb{Z}_p$, and mgEnc can use a random element in $\mathbb{Z}_p^{k+2}$ as a random tape, we can use a random element in $\mathbb{Z}_p^{2m+k+8}$ as a random tape of MstEnc. Due to the homomorphism of iFE and mgFE, for all $N \in \mathbb{N}$, $i \in [n]$, $a_1, \ldots, a_N \in \mathbb{Z}_p$ s.t. $\sum_{j\in[N]} a_j = 1$, $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{Z}_p^m$, $\mathbf{r}_1, \ldots, \mathbf{r}_N \in \mathbb{Z}_p^{2m+k+8}$, we have the following homomorphism of MstEnc:

$$\sum_{j\in[N]} a_j \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{x}; \mathbf{r}_j) = \mathsf{MstEnc}(\mathsf{MSK}, i, \sum_{j\in[N]} a_j \mathbf{x}_j; \sum_{j\in[N]} a_j \mathbf{r}_j).$$

Parse $\mathsf{EK}_i = (\{\mathsf{MCT}_{1,i,j}\}_{i\in[n],j\in[m]}, \{\mathsf{MCT}_{0,i,j}\}_{i\in[n],j\in[D]})$ and let $\mathbf{r}_{b,i,j} \in \mathbb{Z}_p^{2m+k+8}$ be the random tape used to generate $\mathsf{MCT}_{b,i,j}$ for $b \in \{0,1\}$, $i \in [n]$, $j \in [D]$. In other words,

$$\mathsf{MCT}_{b,i,j} = \begin{cases} \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{e}_j); \mathbf{r}_{b,i,j}) & b = 1 \\ \mathsf{MstEnc}(\mathsf{MSK}, i, \mathbf{0}_m); \mathbf{r}_{b,i,j}) & b = 0 \end{cases}$$

From the homomorphism of MstEnc and the fact that Enc can use $\gamma = (\gamma_1, \ldots, \gamma_{(D-1)/2}) \in \mathbb{Z}_p^{(D-1)/2}$ for a random tape, we have

$$\mathsf{Enc}(\mathsf{EK}_i, \mathbf{x} : \gamma) = \mathsf{MstEnc}(\mathsf{MSK}, i, \sum_{j\in[m]} \mathbf{x}[j]\mathbf{e}_j; \mathbf{r})$$

where

$$\mathbf{r} = \sum_{j\in[m]} \mathbf{x}_i[j]\mathbf{r}_{1,i,j} - \left(\sum_{j\in[m]} \mathbf{x}_i[j] - 1\right)\mathbf{r}_{0,i,1} + \sum_{j\in[(D-1)/2]} \gamma_j(\mathbf{r}_{0,i,2j} - \mathbf{r}_{0,i,2j+1}). \tag{7}$$

Here, we use the equality: $\sum_{j\in[m]} \mathbf{x}_i[j] - \left(\sum_{j\in[m]} \mathbf{x}_i[j] - 1\right) + \sum_{j\in[(D-1)/2]}(\gamma_j - \gamma_j) = 1$. Hence, to prove the lemma, it suffices to show that the following distributions are statistically close for all $i \in [n]$:

$$\left\{(\mathbf{r}, \{\mathbf{r}_{1,i,j}\}_{j\in[m]}, \{\mathbf{r}_{0,i,j}\}_{j\in[D]}) : \begin{array}{l}\forall(b,i,j), \mathbf{r}_{b,i,j} \leftarrow \mathbb{Z}_p^{2m+k+8},\ \gamma \leftarrow \mathbb{Z}_p^{(D-1)/2} \\ \mathbf{r} \text{ is defined as Eq. (7)}\end{array}\right\}$$

and

$$\left\{(\mathbf{r}, \{\mathbf{r}_{1,i,j}\}_{j\in[m]}, \{\mathbf{r}_{0,i,j}\}_{j\in[D]}) : \forall(b,i,j), \mathbf{r}_{b,i,j} \leftarrow \mathbb{Z}_p,\ \mathbf{r} \leftarrow \mathbb{Z}_p^{2m+k+8}\right\}$$

This can be shown as follows. For all $j \in [(D-1)/2]$, $\widetilde{\mathbf{r}}_j = \mathbf{r}_{0,i,2j} - \mathbf{r}_{0,i,2j+1}$ is uniformly distributed in $\mathbb{Z}_p^{2m+k+8}$, and thus $\widetilde{\mathbf{r}}_1, \ldots, \widetilde{\mathbf{r}}_{(D-1)/2}$ span $\mathbb{Z}_p^{2m+k+8}$ with overwhelming probability if $(D-1)/2 \geq 2m+k+8$. Hence, $\sum_{j\in[(D-1)/2]} \gamma_j(\mathbf{r}_{0,i,2j} - \mathbf{r}_{0,i,2j+1}) = \sum_{j\in[(D-1)/2]} \gamma_j\widetilde{\mathbf{r}}_j$ is randomly distributed even given $(\{\mathbf{r}_{1,i,j}\}_{j\in[m]}, \{\mathbf{r}_{0,i,j}\}_{j\in[D]})$. This concludes the proof.

Lemma 5.3. *For all PPT adversaries $\mathcal{A}$, there exists a PPT adversary $\mathcal{B}$ against* qcFE *in Sect. 4 such that* $|\mathsf{P}(\mathcal{A}, \mathsf{H}_1^\beta) - \mathsf{P}(\mathcal{A}, \mathsf{H}_\mathsf{f}^\beta)| \leq \mathsf{Adv}_\mathcal{B}^{\mathsf{qcFE}}(\lambda)$.

Proof. The difference of these hybrids is whether CT_i^μ is encryption of $\mathbf{x}_i^{\mu,\beta}$ or $\mathbf{x}_i^{\mu,0}$. We can construct $\mathcal{B}$ as follows.

1. $\mathcal{B}$ is given qcPP and gives it to $\mathcal{A}$.
2. $\mathcal{A}$ outputs $(\mathcal{CS}, \{i, \mathbf{x}_i^{\mu,0}, \mathbf{x}_i^{\mu,1}\}_{i\in[n],\mu\in[q_c]}, \mathcal{FS} = \{\mathbf{c}^\nu\}_{\nu\in[q_k]})$, and $\mathcal{B}$ chooses $\beta \leftarrow \{0,1\}$ and queries its own oracle on $\mathcal{MS}, \mathcal{FS}$ where
$$\mathcal{MS} = \begin{pmatrix} \{i, \mathsf{lab}_0, \mathbf{e}_j, \mathbf{e}_j\}_{i\in\mathcal{CS},j\in[m]}, \{(i, \mathsf{lab}_0, \mathbf{0}_m, \mathbf{0}_m) \times D\}_{i\in\mathcal{CS}} \\ \{i, \mathsf{lab}_0, \mathbf{x}_i^{\mu,\beta}, \mathbf{x}_i^{\mu,0}\}_{i\in[n],\mu\in[q_c]} \end{pmatrix}.$$
3. $\mathcal{B}$ is given
$$(\{\mathsf{cCT}_{1,i,j}\}_{i\in\mathcal{CS},j\in[m]}, \{\mathsf{cCT}_{0,i,j}\}_{i\in\mathcal{CS},j\in[D]}, \{\mathsf{cCT}_i^\mu\}_{i\in[n],\mu\in[q_c]}, \{\mathsf{cSK}^\nu\}_{\nu\in[q_k]})$$
where $\mathsf{cCT}_{1,i,j}, \mathsf{cCT}_{0,i,j}, \mathsf{cCT}_i^\mu$ are ciphertexts of qcFE for $(i, \mathsf{lab}_0, \mathbf{e}_j)$, $(i, \mathsf{lab}_0, \mathbf{0}_m)$, $(i, \mathsf{lab}_0, \mathbf{x}_i^{\mu,\beta/0})$, respectively, and gives it to $\mathcal{A}$ by setting $\mathsf{EK}_i = (\{\mathsf{cCT}_{1,i,j}\}_{j\in[m]}, \{\mathsf{cCT}_{0,i,j}\}_{j\in[D]})$.
4. $\mathcal{A}$ outputs β', and $\mathcal{B}$ outputs β' as it is.

We can confirm the above simulation of $\mathcal{B}$ is valid from the three observations. First, $\mathcal{B}$'s query satisfies the game condition (recall that the adversary can query any pair of the same messages for corrupted slot). Second, $\mathsf{qcPP} = \mathsf{iPP}^{(1)} = \mathbb{G}$ where qcPP is the public parameter of qcFE. Third, $\mathsf{MstEnc}(\mathsf{MSK}, \cdot, \cdot)$ and $\mathsf{qcEnc}(\mathsf{cMSK}, \cdot, \mathsf{lab}_0, \cdot)$ (the encryption algorithm of qcFE) are the exactly the same.

References

1. Abdalla, M., Benhamouda, F., Gay, R.: From single-input to multi-client inner-product functional encryption. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 552–582. Springer, Heidelberg (2019)
2. Abdalla, M., Benhamouda, F., Kohlweiss, M., Waldner, H.: Decentralizing inner-product functional encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 128–157. Springer, Heidelberg (2019)
3. Abdalla, M., Bourse, F., De Caro, A., Pointcheval, D.: Simple functional encryption schemes for inner products. In: Katz, J. (ed.) PKC 2015. LNCS, vol. 9020, pp. 733–751. Springer, Heidelberg (2015)
4. Abdalla, M., Catalano, D., Fiore, D., Gay, R., Ursu, B.: Multi-input functional encryption for inner products: function-hiding realizations and constructions without pairings. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part I. LNCS, vol. 10991, pp. 597–627. Springer, Heidelberg (2018)
5. Abdalla, M., Gay, R., Raykova, M., Wee, H.: Multi-input inner-product functional encryption from pairings. In: Coron, J.S., Nielsen, J.B. (eds.) EUROCRYPT 2017, Part I. LNCS, vol. 10210, pp. 601–626. Springer, Heidelberg (2017)
6. Abdalla, M., Gong, J., Wee, H.: Functional encryption for attribute-weighted sums from k-Lin. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 685–716. Springer, Heidelberg (2020)
7. Agrawal, S., Koppula, V., Waters, B.: Impossibility of simulation secure functional encryption even with random oracles. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018. LNCS, vol. 11239, pp. 659–688. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03807-6_24
8. Agrawal, S., Gorbunov, S., Vaikuntanathan, V., Wee, H.: Functional encryption: new perspectives and lower bounds. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 500–518. Springer, Heidelberg (2013)
9. Agrawal, S., Goyal, R., Tomida, J.: Multi-input quadratic functional encryption from pairings. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part IV. LNCS, vol. 12828, pp. 208–238. Springer, Heidelberg, Virtual Event (2021)
10. Agrawal, S., Goyal, R., Tomida, J.: Multi-party functional encryption. In: Nissim, K., Waters, B. (eds.) TCC 2021. LNCS, vol. 13043, pp. 224–255. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90453-1_8
11. Agrawal, S., Maitra, M.: FE and iO for turing machines from minimal assumptions. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 473–512. Springer, Heidelberg (2018)
12. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M.J.B. (eds.) Part I. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015)
13. Ananth, P., Jain, A.: Indistinguishability obfuscation from compact functional encryption. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 308–326. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47989-6_15
14. Baltico, C.E.Z., Catalano, D., Fiore, D., Gay, R.: Practical functional encryption for quadratic functions with applications to predicate encryption. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 67–98. Springer, Heidelberg (2017)
15. Bishop, A., Jain, A., Kowalczyk, L.: Function-hiding inner product encryption. In: Iwata, T., Cheon, J.H. (eds.) ASIACRYPT 2015. LNCS, vol. 9452, pp. 470–491. Springer, Heidelberg (2015)

16. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: Guruswami, V. (ed.) 56th FOCS, pp. 171–190. IEEE Computer Society Press (2015)
17. Bitansky, N., Vaikuntanathan, V.: Indistinguishability obfuscation from functional encryption. In: IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17–20 October, 2015, pp. 171–190 (2015)
18. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011)
19. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_29
20. Brakerski, Z., Segev, G.: Function-private functional encryption in the private-key setting. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part II. LNCS, vol. 9015, pp. 306–324. Springer, Heidelberg (Mar (2015)
21. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Decentralized multi-client functional encryption for inner product. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part II. LNCS, vol. 11273, pp. 703–732. Springer, Heidelberg (2018)
22. Chotard, J., Dufour Sans, E., Gay, R., Phan, D.H., Pointcheval, D.: Multi-client functional encryption with repetition for inner product. Cryptology ePrint Archive, Report 2018/1021 (2018). https://eprint.iacr.org/2018/1021
23. Datta, P., Okamoto, T., Tomida, J.: Full-hiding (unbounded) multi-input inner product functional encryption from the k-Linear assumption. In: Abdalla, M., Dahab, R. (eds.) PKC 2018. LNCS, vol. 10770, pp. 245–277. Springer, Heidelberg (2018)
24. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: 54th FOCS, pp. 40–49. IEEE Computer Society Press (2013)
25. Garg, S., Gentry, C., Halevi, S., Zhandry, M.: Functional encryption without obfuscation. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 480–511. Springer, Heidelberg (2016)
26. Goldwasser, S., et al.: Multi-input functional encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 578–602. Springer, Heidelberg (2014)
27. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from well-founded assumptions. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 60–73 (2021)
28. Jain, A., Lin, H., Sahai, A.: Indistinguishability obfuscation from LPN over f_p, dlin, and prgs in nĉ 0. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology EUROCRYPT 2022. Lecture Notes in Computer Science, vol. 13275, pp. 670–690. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06944-4_23
29. Libert, B., Titiu, R.: Multi-client functional encryption for linear functions in the standard model from LWE. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part III. LNCS, vol. 11923, pp. 520–551. Springer, Heidelberg (2019)
30. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556 (2010). https://eprint.iacr.org/2010/556
31. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005). https://doi.org/10.1007/11426639_27

32. Tomida, J.: Tightly secure inner product functional encryption: multi-input and function-hiding constructions. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019. LNCS, vol. 11923, pp. 459–488. Springer, Heidelberg (2019)
33. Tomida, J., Abe, M., Okamoto, T.: Efficient inner product functional encryption with full-hiding security. IEICE Trans. Fundam. Electron. Commun. Comput. Sci. **103**-A(1), 33–40 (2020)
34. Tomida, J., Takashima, K.: Unbounded inner product functional encryption from bilinear maps. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018. LNCS, vol. 11273, pp. 609–639. Springer, Heidelberg (2018)