Unbounded Quadratic Functional Encryption and More from Pairings

Junichi Tomida

NTT Social Informatics Laboratories, Japan
junichi.tomida.vw@hco.ntt.co.jp

Abstract. We propose the first unbounded functional encryption (FE) scheme for quadratic functions and its extension, in which the sizes of messages to be encrypted are not a priori bounded. Prior to our work, all FE schemes for quadratic functions are bounded, meaning that the message length is fixed at the setup. In the first scheme, encryption takes $\{x_i\}_{i\in S_c}$, key generation takes $\{c_{i,j}\}_{i,j\in S_k}$, and decryption outputs $\sum_{i,j\in S_k} c_{i,j}x_ix_j$ if and only if $S_k \subseteq S_c$, where the sizes of S_c and S_k can be arbitrary. Our second scheme is the extension of the first scheme to partially-hiding FE that computes an arithmetic branching program on a public input and a quadratic function on a private input. Concretely, encryption takes a public input $\mathbf{u}$ in addition to $\{x_i\}_{i\in S_c}$, a secret key is associated with arithmetic branching programs $\{f_{i,j}\}_{i,j\in S_k}$, and decryption yields $\sum_{i,j\in S_k} f_{i,j}(\mathbf{u})x_ix_j$ if and only if $S_k \subseteq S_c$. Both our schemes are based on pairings and secure in the simulation-based model under the standard MDDH assumption.

Keywords: functional encryption, unbounded, quadratic functions, arithmetic branching programs, pairings

Table of Contents

1 Introduction

Functional encryption (FE) [O'N10, BSW11] is a new cryptographic paradigm that allows a decrypter to learn a function value of the underlying message without revealing any other information and enables fine-grained access control over encrypted data. This is in contrast to traditional public-key encryption, which only provides all-or-nothing decryption. Concretely, an FE scheme that supports a function class $\mathcal{F}$ allows an owner of a master secret to issue a secret key SK for a function $f \in \mathcal{F}$. Decryption of a ciphertext CT for a message x with SK yields $f(x)$ and nothing else. Functional encryption has been extensively studied in the literature, with elegant constructions supporting various function classes, achieving different notions of security and from various assumptions, e.g., [GGH$^+$13b, GGHZ16, BS15, ABDP15, BCFG17].

In this paper, we focus on the following FE system. Consider a database consisting of pairs of a unique public identifier i and an encrypted private attribute x_i (e.g., age, medical history, salary, etc.). An authority can issue a secret key SK that allows a user to compute an analysis f' using *a portion of* the encrypted data with respect to some identifier set S_k. In other words, the user given SK can learn $f'(\{x_i\}_{i \in S_k})$ if and only if $S_k \subseteq S_c$ from the encrypted database, where S_c is the set of all identifiers in the database. We consider that preventing decryption in the case $S_k \not\subseteq S_c$ is important since otherwise the decrypter may learn specific information on some private attribute, which is undesirable in many applications (for instance, even in the case where S_k is large and f' computes average, the decrypter can learn exact x_i if $S_k \cap S_c = \{i\}$). In both theory and practice, it is arguably desirable if the system satisfies the following properties:

1. the size of the database that can be encrypted is not a priori bounded;
2. the size of the encrypted database is linear in the number of records $|S_c|$; and
3. the system is based on standard assumption and does not rely on heavy cryptographic tools such as obfuscation [GGH$^+$13b] and multi-linear maps [GGH13a].

Most of the existing FE schemes do not satisfy item 1 since the size of messages to be encrypted is a priori fixed. To our knowledge, the exceptions are FE for Turing machines [BCP14, IPS15, AS16], unbounded FE for inner product [TT18, DP19], and FE for attribute-weighted sums [AGW20, DP21]. However, since all the FE schemes for Turing machines (secure against unbounded collusion) rely on obfuscation, only a few FE schemes satisfy all the properties simultaneously. Furthermore, the output of the functions in these few FE schemes are all linear in $\{x_i\}_{i \in S_c}$. This naturally motivates the following question:

Can we construct an FE scheme for quadratic functions with all the properties?

We basically use the term "unbounded" to describe the property of item 1, but crucially, it also implies that the system supports variable-length plaintext. Note that most FE schemes support only fixed-length plaintext, meaning that we always have $S_c = S_k = [n]$ for a fixed polynomial n. In fixed-length schemes, when encrypting messages shorter than the fixed length, it is necessary to do something like zero padding, and it is impossible to encrypt messages longer than the fixed length.

From an efficiency standpoint, the variable-length property is quite important in systems that may handle data of various lengths. Let us consider a case where a country introduces an FE system, and local governments use it to encrypt the database of their residents. It is natural for the number of residents in each district to be various sizes. At some point, local governments may annex their regions, and the population of the new region would exceed the system limit. In such a case, we have to re-deploy the encryption system with a larger limit if they are using a fixed-length FE scheme. This problem can be avoided by setting the system limit with a huge margin in the setup phase. However, this solution brings a significant overhead to the system since the lengths of all ciphertexts become at least linear in the fixed system limit even if most plaintexts to be encrypted in the system are much shorter than the fixed length!

In contrast, the ciphertext sizes of variable-length FE schemes are linear in the size $|S_c|$ of each database as specified in item 2. Hence, variable-length FE schemes can be much more efficient than fixed-length FE schemes in situations as described above. Furthermore, we do not need to care even the system limit if we can use an *unbounded* FE scheme. However, all previous FE schemes for quadratic functions are fixed-length [Lin17, BCFG17, RPB$^+$19, Gay20, Wee20, GQ20, AGT21], and no unbounded (or even no variable-length) schemes are known. Hence, the above question is not only of theoretical interest but also important from a practical viewpoint.

1.1 Our Results

We construct an unbounded (public-key) FE scheme for quadratic functions and its extension. Both schemes have semi-adaptive, simulation-based security under the matrix decisional Diffie-Hellman (MDDH) assumption in the random oracle model (ROM) and thus satisfy the three properties simultaneously. Note that achieving adaptive security in FE for quadratic functions is a long-standing open problem, and no quadratic FE scheme achieves adaptive security (except the scheme based on the generic group model). We also remark that we cannot use the ROM straightforwardly to extend the existing quadratic FE schemes to be unbounded, and we overcome many hurdles to obtain the current results. We elaborate on this later in the technical overview. We leave constructing unbounded quadratic FEs without the ROM as an interesting open problem.

The first scheme is unbounded FE for quadratic functions, that is, f' in the above context can be any quadratic function. More formally, the message space and the function space is specified as $\mathcal{X} = \{(x_1, x_2) \in 2^{[p]} \times \bigcup_{i\in[p]} \mathbb{Z}_p^i \mid |x_1| = |x_2|\}$, and $\mathcal{F} = \{(f_1, f_2) \in 2^{[p]} \times \bigcup_{i\in[p]} \mathbb{Z}_p^{i^2} \mid |f_1|^2 = |f_2|\}$, respectively, where p is an exponentially large prime[1], and $2^{[p]}$ denotes the set consisting of all subset of

[1] Concretely, p is an order of bilinear groups that the scheme based on.

Scheme	$\|\mathsf{PK}\|$	$\|\mathsf{CT}\|$	$\|\mathsf{SK}\|$	Variable-length	Unbounded	w/o RO
Fixed-length schemes	$O(n)$	$O(n)$	$O(n)$ or $O(1)$	×	×	✓
Ours (bounded)	$O(n')$	$O(\|S_c\|)$	$O(\|S_k\|)$	✓	×	✓
Ours (unbounded)	$O(1)$	$O(\|S_c\|)$	$O(\|S_k\|)$	✓	✓	×

Table 1. Comparison among public-key functional encryption schemes for quadratic functions. Fixed-length schemes refer to [BCFG17, RPB+19, Gay20, Wee20, GQ20, AGT21]. In this table, n is the fixed vector length, S_c and S_k are the identifier sets, and n' is the upper bound of the vector length, i.e., S_c and S_k must be subsets of $[n']$ in the bounded scheme. RO stands for random oracles.

$[p]$. For $x = (S_c, \{x_i\}_{i \in S_c}) \in \mathcal{X}$ and $f = (S_k, \{c_{i,j}\}_{i,j \in S_k}) \in \mathcal{F}$, $f(x)$ is defined as

$$f(x) = \begin{cases} \sum_{i,j \in S_k} c_{i,j} x_i x_j & S_k \subseteq S_c \\ \perp & \text{otherwise} \end{cases}$$

where S_c is clear in the ciphertext. Observe that S_c can be an arbitrary subset of $[p]$ where p is an exponentially large prime, and thus the size of S_c is unbounded since encryption is a polynomial time algorithm.

Our unbounded quadratic FE scheme can be easily modified to a (bounded) variable-length quadratic FE scheme *without* random oracles. In the scheme, S_c and S_k must be subsets of a fixed poly-sized set $[n']$ instead of an exponentially large set $[p]$. We present a comparison of our quadratic FE schemes with previous schemes in Table 1.

The second scheme is inspired by the recent works of partially-hiding functional encryption [JLMS19, AJL+19, Wee20, GJLS21], where a message consists of public input $\mathbf{u}$ and private input $\mathbf{x}$ while a secret key is associated with f' in NC1 or arithmetic branching programs (ABPs), and decryption yields $f(\mathbf{u}, \mathbf{x}) = \langle f'(\mathbf{u}), \mathbf{x} \otimes \mathbf{x} \rangle$. We extend this functionality to unbounded FE for quadratic functions. Assume that each database additionally has a public input $\mathbf{u}$ (e.g., the description of the database) with a fixed length n, while a secret key is associated with S_k and arithmetic branching program f'[2] the input and output lengths of which are n and $|S_k|^2$, respectively. Then, the decryption reveals $\sum_{i,j \in S_k} f'_{i,j}(\mathbf{u}) x_i x_j$ where $f'_{i,j}(\mathbf{u})$ is the (i,j)-th output of $f'(\mathbf{u})$. Formally, the message space and the function space is specified as $\mathcal{X} = \{(x_1, x_2, x_3) \in \mathbb{Z}_p^n \times 2^{[q]} \times \bigcup_{i \in [q]} \mathbb{Z}_p^i \mid |x_2| = |x_3|\}$, and $\mathcal{F} = \{(f_1, f_2) \in 2^{[q]} \times \bigcup_{i \in [q]} \mathcal{F}^{\mathsf{ABP}}_{n,i^2} \mid |f_1|^2 = \mathsf{OutLen}(f_2)\}$, respectively, where $q \in \mathbb{N}$ is an exponentially large number ($q = p - 1$ in our scheme), $\mathcal{F}^{\mathsf{ABP}}_{n,n'}$ denotes the set of all ABPs with the input and output lengths being n and n', respectively, and $\mathsf{OutLen}(f_2)$ denotes the output length of f_2. For $x = (\mathbf{u}, S_c, \{x_i\}_{i \in S_c}) \in \mathcal{X}$ and $f = (S_k, f') \in \mathcal{F}$, $f(x)$ is defined as

$$f(x) = \begin{cases} \sum_{i,j \in S_k} f'_{i,j}(\mathbf{u}) x_i x_j & S_k \subseteq S_c \\ \perp & \text{otherwise} \end{cases}$$

where $\mathbf{u}, S_c$ are clear in the ciphertext. We call this functionality ABP $\circ$ UQF.

[2] Note that ABPs are a stronger computational model than NC1 circuits.

By similar observation to [AGW20], we can confirm that FE for ABP $\circ$ UQF subsumes many classes of FE: (unbounded) FE for inner product [ABDP15, TT18]; FE for quadratic functions [BCFG17]; attribute-based encryption for ABPs [LL20]; attribute-based inner product FE [ACGU20]; and attribute-based quadratic FE [Wee20] as well as unbounded FE for quadratic functions (our first scheme)[3]. Hence, for instance, FE for ABP $\circ$ UQF allows the decryption of an encrypted database with description $\mathbf{u}$ and identifier set S_c in which it first checks whether $\mathbf{u}$ satisfies a NC1 predicate P and then outputs a quadratic function f' over the portion $\{x_i\}_{i \in S_k}$ of the private input of the database iff $\mathsf{P}(\mathbf{u}) = 1$ and $S_k \subseteq S_c$, because such computation can be expressed by ABPs.

Comparison with FE for attribute-weighted sums. Although FE for ABP $\circ$ UQF is similar to FE for attribute-weighted sums [AGW20] in that they can encrypt a database with unbounded length, and a secret key is associated with an ABP, their functionalities are essentially different as follows. The public input $\mathbf{u}$ is specific to a database in FE for ABP $\circ$ UQF while each record has the public input $\mathbf{u}_i$ in FE for attribute-weighted sums. In decryption with a secret key for an ABP f, the output of FE for ABP $\circ$ UQF is the weighted-sum of $x_i x_j$ for $i, j \in S_k$ with the weight being $f_{i,j}(\mathbf{u})$ while that of FE for attribute-weighted sums is the weighted-sum of x_i for $i \in S_c$ with the weight being $f(\mathbf{u}_i)$.

1.2 Technical Overview

For simplicity, we stick to the case using the SXDH assumption, which is the special case of the MDDH assumption in this overview.

Why the ROM does not work straightforwardly? Before diving into our construction, we first see why it is difficult to extend the existing quadratic FE schemes to be unbounded by the ROM. For all public-key quadratic FE schemes [BCFG17, RPB$^+$19, Gay20, Wee20, GQ20], a public key PK and a secret key SK for any quadratic function f consist of following elements:

$$\mathsf{PK} = ([\mathbf{A}_1]_1, [\mathbf{A}_2]_2, [\mathbf{B}]_1, \ldots), \quad \mathsf{SK} = ([\mathbf{D}]_i, \ldots)$$

where $\mathbf{A}_1, \mathbf{A}_2$ are (pseudo)random matrices in $\mathbb{Z}_p$ the sizes of which depend on the message length m, $\mathbf{B}, \mathbf{D}$ are some matrices in $\mathbb{Z}_p$, $i \in \{1, 2\}$, and $[\cdot]_i$ denotes element-wise exponentiation in the source group G_i. How to define these matrices and i depends on the scheme. The natural idea to make the scheme unbounded is to generate $[\mathbf{A}_1]_1, [\mathbf{A}_2]_2$ by hash functions $H_1 : \{0,1\}^* \to G_1$ and $H_2 : \{0,1\}^* \to G_2$ in an ad hoc way in encryption. In all the existing schemes, however, either $\mathbf{B}$ [Gay20] or $\mathbf{D}$ [BCFG17, RPB$^+$19, GQ20, Wee20] contains the entries of the form $va_1a_2 + c$, where a_1, a_2 are entries of $\mathbf{A}_1, \mathbf{A}_2$, respectively, and v, c are $\mathbb{Z}_p$ elements that are independent of both a_1 and a_2. It is not hard to see that neither $[va_1a_2 + c]_1$ nor $[va_1a_2 + c]_2$ can be computed efficiently even in symmetric pairings. Hence, this strategy makes encryption or key generation

[3] This does not mean that our results imply the listed schemes since we ignore the security requirement here and focus on only functionalities.

inefficient. Furthermore, such a construction will not become collusion resistant, that is, a user can generate a secret key for $S_{k,1}$ from secret keys for $S_{k,2}$ and $S_{k,3}$ such that $S_{k,1} \subseteq S_c$ but $S_{k,2}, S_{k,3} \not\subseteq S_c$ in a certain case [DP19].

Starting from Lin's secret-key FE scheme. Since the known public-key quadratic FE schemes are not ROM-friendly as observed, we construct a new public-key quadratic FE scheme that is inspired by the secret-key quadratic FE scheme from pairings by Lin [Lin17]. Her scheme builds on the public-key IPFE scheme from DDH by Abdalla *et al.* [ABDP15] (ABDP), which is described as follows:

$\mathsf{Setup}(1^\lambda)$**:** $\mathbf{w} \leftarrow \mathbb{Z}_p^m,\ \mathsf{PK} = [\mathbf{w}],\ \mathsf{MSK} = \mathbf{w}$.
$\mathsf{Enc}(\mathsf{PK}, \mathbf{x} \in \mathbb{Z}^m)$**:** $s \leftarrow \mathbb{Z}_p,\ \mathsf{CT} = (\mathsf{CT}_1, \mathsf{CT}_2) = ([s], [\mathbf{x} + s\mathbf{w}])$.
$\mathsf{KeyGen}(\mathsf{MSK}, \mathbf{c} \in \mathbb{Z}^m)$**:** $\mathsf{SK} = -\mathbf{c}^\top \mathbf{w}$.
$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** $\mathsf{SKCT}_1 + \mathbf{c}^\top \mathsf{CT}_2 = -\mathbf{c}^\top \mathbf{w}[s] + \mathbf{c}^\top[\mathbf{x} + s\mathbf{w}] = [\langle \mathbf{c}, \mathbf{x} \rangle]$.

Lin's quadratic FE scheme uses a clever interleaving of IPFE schemes. To compress the size of ABDP ciphertexts for quadratic terms, she uses function-hiding IPFE where a secret key hides the underlying vector as well as a ciphertext hides the message [BJK15]. Decryption of components in this scheme yields a ciphertext of the ABDP IPFE scheme, while a secret key of the ABDP scheme is generated using another function-hiding IPFE. Finally, decryption of ABDP IPFE allows to recover the output. In more detail, let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a function-hiding IPFE scheme based on pairings, which outputs a decryption value as an exponent of the target-group generator. Her quadratic FE scheme is informally described as follows (we omit the components of the scheme that are only used in the security proof):

$\mathsf{Setup}(1^\lambda)$**:** $\mathbf{w} = (w_1, \ldots, w_m), \widetilde{\mathbf{w}} = (\widetilde{w}_1, \ldots, \widetilde{w}_m) \leftarrow \mathbb{Z}_p^m,\ \mathsf{iMSK}' \leftarrow \mathsf{iSetup}(1^\lambda)$
  $\mathsf{MSK} = (\mathsf{iMSK}', \mathbf{w}, \widetilde{\mathbf{w}})$.
$\mathsf{Enc}(\mathsf{MSK}, \mathbf{x} \in \mathbb{Z}^m)$**:** $s \leftarrow \mathbb{Z}_p,\ \mathsf{iCT}' \leftarrow \mathsf{iEnc}(\mathsf{iMSK}', s),\ \mathsf{iMSK} \leftarrow \mathsf{iSetup}(1^\lambda)$
  $\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, (x_i, w_i)), \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, (x_i, s\widetilde{w}_i))$.
  $\mathsf{CT} = (\mathsf{iCT}', \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in [m]})$.
$\mathsf{KeyGen}(\mathsf{MSK}, \mathbf{c} = \{c_{i,j}\}_{i,j \in [m]} \in \mathbb{Z}^{m^2})$**:**
  $\mathsf{SK} = \mathsf{iSK}' \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}', \mathbf{c}^\top(\mathbf{w} \otimes \widetilde{\mathbf{w}}))$.
$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** $\sum_{i,j \in [m]} c_{i,j} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j) - \mathsf{iDec}(\mathsf{iCT}', \mathsf{iSK}') = [\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle]_T$.

In decryption, we compute $\mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j) = [x_i x_j + s w_i \widetilde{w}_j]_T$, which can be seen as the (i,j)-th element of the ABDP ciphertext $[\mathbf{x} \otimes \mathbf{x} + s\mathbf{w} \otimes \widetilde{\mathbf{w}}]_T$, and $-\mathsf{iDec}(\mathsf{iCT}', \mathsf{iSK}') = [-s\mathbf{c}^\top(\mathbf{w} \otimes \widetilde{\mathbf{w}})]_T$, where $-\mathbf{c}^\top(\mathbf{w} \otimes \widetilde{\mathbf{w}})$ is an ABDP secret key for $\mathbf{c}$. Since $\mathbf{w} \otimes \widetilde{\mathbf{w}}$ only appears on the exponent, it looks uniformly distributed under the SXDH assumption.

Making Lin's scheme public-key. We next show how to turn her scheme into a public-key scheme. Observe that her scheme is secret-key since it uses the function-hiding property of the secret-key IPFE. More specifically, encryption chooses fresh iMSK by itself while iMSK' is the part of MSK. This means that we would be able to make her scheme public-key if we can publicly encrypt s into

iCT' in encryption, and at the same time, iSK' is still function-hiding so that the security proof goes well.

Fortunately, we already have slotted IPFE [LV16], which is a hybrid between public-key IPFE and a function-hiding IPFE and satisfies the above properties. Specifically, both message and key spaces of slotted IPFE are separated into two slots $\mathbb{Z}_p^{m_1}$ and $\mathbb{Z}_p^{m_2}$, and we can publicly encrypt all messages of the form $(\mathbf{x}_1, \mathbf{0})$ for $\mathbf{x}_1 \in \mathbb{Z}_p^{m_1}$ via slot encryption algorithm $\mathsf{iSlotEnc}$ while we need a master secret key to encrypt a message of the form $(\mathbf{x}_1, \mathbf{x}_2) \in \mathbb{Z}_p^{m_1} \times \mathbb{Z}_p^{m_2}$ for $\mathbf{x}_2 \neq \mathbf{0}$ via encryption algorithm iEnc. A secret key for $(\mathbf{y}_1, \mathbf{y}_2) \in \mathbb{Z}_p^{m_1} \times \mathbb{Z}_p^{m_2}$ is function-hiding with respect to $\mathbf{y}_2$, which is essential for the security proof.

Another nice property of (slotted) IPFE is that (slot) encryption and key generation can take a group element in G_1 and G_2 of pairing groups as input, respectively [LL20]. Thus, we can publish $[\mathbf{w}]_1$ and $[\widetilde{\mathbf{w}}]_2$ as a part of public key and use them to generate $\mathsf{iCT}_i, \mathsf{iSK}_i$ in encryption. It seems that the modified scheme is now public-key, but unfortunately, this is not the case. This is because, in the security proof of Lin's scheme, we argue that $[w_i \widetilde{\mathbf{w}}]_2$ looks random given PK, but it is not the case if $[\mathbf{w}]_1$ is included in PK. To circumvent this problem, we modify Lin's scheme to obtain a public-key scheme using a slotted IPFE scheme $\mathsf{iFE}' = (\mathsf{iSetup}', \mathsf{iSlotEnc}', \mathsf{iEnc}', \mathsf{iKeyGen}', \mathsf{iDec}')$ as follows (we again omit the components of the scheme that are only required for the proof of security):

$\mathsf{Setup}(1^\lambda)$**:** $\mathbf{w} = (w_1, \ldots, w_m) \leftarrow \mathbb{Z}_p^m,\ (\mathsf{iPK}', \mathsf{iMSK}') \leftarrow \mathsf{iSetup}'(1^\lambda)$
  $\mathsf{PK} = ([\mathbf{w}]_2, \mathsf{iPK}'),\ \mathsf{MSK} = \mathsf{iMSK}'.$
$\mathsf{Enc}(\mathsf{PK}, \mathbf{x} \in \mathbb{Z}^m)$**:** $\mathbf{s} = (s_1, \ldots, s_m) \leftarrow \mathbb{Z}_p^m,\ \mathsf{iCT}' \leftarrow \mathsf{iSlotEnc}'(\mathsf{iPK}', [\mathbf{s}]_1)$
  $\mathsf{iMSK} \leftarrow \mathsf{iSetup}(1^\lambda)$
  $\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [(x_i, s_i)]_1),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [(x_i, w_i)]_2).$
  $\mathsf{CT} = (\mathsf{iCT}', \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in [m]}).$
$\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, \mathbf{c} = \{c_{i,j}\}_{i,j \in [m]} \in \mathbb{Z}^{m^2})$**:**
  $\mathsf{SK} = \mathsf{iSK}' \leftarrow \mathsf{iKeyGen}'(\mathsf{iMSK}', [(\sum_{j \in [m]} c_{1,j} w_j, \ldots, \sum_{j \in [m]} c_{m,j} w_j)]_2).$
$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** $\sum_{i,j \in [m]} c_{i,j} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j) - \mathsf{iDec}'(\mathsf{iCT}', \mathsf{iSK}') = [\langle \mathbf{c}, \mathbf{x} \otimes \mathbf{x} \rangle]_T.$

The above issue does not occur in this modified scheme, that is, we can argue that $[s_i \mathbf{w}]_2$ looks random under the SXDH assumption even if PK is given. Even better, this scheme is ROM-friendly in a sense that Enc and KeyGen are still efficient even if $[\mathbf{w}]_2$ is generated by hashing as $[w_i]_2 = H(i)$! Note that the ciphertext size of the above scheme is still linear in m since the ciphertext size of the slotted IPFE scheme is linear in m_1 and m_2, and $m_2 = 1$ is sufficient for the security proof.

How to achieve the partial decryption. As discussed above, our goal is to allow an owner of a secret key with respect to S_k to decrypt the portion S_k of a ciphertext for S_c if and only if $S_k \subseteq S_c$. Our observation is that if the underlying slotted IPFE scheme iFE' is unbounded and allows the partial decryption, the entire quadratic FE scheme is also unbounded and allows the partial decryption. Intuitively, $\{\mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j)\}_{i,j \in S_c}$ in CT reveals only $\{[x_i x_j + s_i w_j]_T\}_{i,j \in S_c}$, and $\{[s_i w_j]_T\}_{i,j \in S_c}$ looks random under the SXDH assumption. Therefore, the decrypter can learn $[\sum_{i,j \in S_k} c_{i,j} x_i x_j]_T$ if and only if it can compute $[\sum_{i,j \in S_k} c_{i,j} s_i w_j]_T$. This is why the decryption condition of the quadratic FE scheme is reduced to that of the underlying slotted IPFE scheme. Thus, the remaining task is to construct an unbounded IPFE that allows the partial decryption armed with the *slotted* property, which is necessary to achieve simulation-based security of our unbounded quadratic FE schemes.[4]

The closest scheme to what we need is the public-key unbounded IPFE scheme by Tomida and Takashima [TT18], which is an unbounded IPFE allowing the partial decryption. However, their scheme is deficient in the two points. First, it is not slotted. Second, it can encode only a $\mathbb{Z}_p$ element for each identifier while we need to encode *a vector* consisting of group elements for each identifier in encryption and key generation[5]. This is why we construct a new unbounded slotted IPFE scheme, which is of independent interest. Recall that their scheme is a direct construction based on the DPVS framework [OT10], and its security analysis is rather complex. In contrast, our scheme is generically obtained from slotted IPFE and thus much simpler.

We construct the unbounded slotted IPFE (slotted uIPFE) scheme in two steps. First, we construct a predicate slotted IPFE (slotted pIPFE) from a slotted IPFE, which is a slotted variant of the predicate IPFE proposed in [AGT21]. Then, we construct a slotted uIPFE from a slotted pIPFE.

Slotted pIPFE is an extension of slotted IPFE in which we can control decryption conditions by an inner product predicate. Specifically, the message space is separated in two slots $\mathbb{Z}_p^d \times G_1^{m_1}$ and $G_1^{m_2}$, and we can publicly encrypt all messages of the form $(\mathbf{u}, [\mathbf{x}_1]_1, [\mathbf{0}]_1)$ for $(\mathbf{u}, [\mathbf{x}_1]_1) \in \mathbb{Z}_p^d \times G_1^{m_1}$ while we need the master secret key to encrypt a message of the form $(\mathbf{u}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1)$ for $\mathbf{x}_2 \neq \mathbf{0}$. A secret key for $(\mathbf{v}, [\mathbf{y}_1]_2, [\mathbf{y}_2]_2) \in \mathbb{Z}_p^d \times G_2^{m_1} \times G_2^{m_2}$ is function-hiding with respect to $[\mathbf{y}_2]_2$, and decryption of these reveals $[\langle (\mathbf{x}_1, \mathbf{x}_2), (\mathbf{y}_1, \mathbf{y}_2) \rangle]_T$ if and only if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$. The construction is almost the same as pIPFE in [AGT21] except that we use a slotted IPFE as a building block instead of an IPFE.

We next define slotted uIPFE more formally. The message space consists of two slots $\{(x_1, x_2) \in 2^{[p]} \times \bigcup_{i \in [p]} (G_1^{m_1})^i \mid |x_1| = |x_2|/m_1\}$ and $G_1^{m_2}$, and we can publicly encrypt all messages of the form $(S_c, \{[\mathbf{x}_i]_1\}_{i \in S_c}, [\mathbf{0}]_1)$ while we need a master secret key to encrypt of the form $(S_c, \{[\mathbf{x}_i]_1\}_{i \in S_c}, [\mathbf{x}_0]_1)$ for $\mathbf{x}_0 \neq \mathbf{0}$ similarly to the other slotted FE schemes. A secret key for $(S_k, \{[\mathbf{y}_i]_2\}_{i \in S_k}, [\mathbf{y}_0]_2)$ is function-hiding with respect to $[\mathbf{y}_0]_2$, and decryption reveals $[\sum_{i \in S_k} \langle \mathbf{x}_i, \mathbf{y}_i \rangle + \langle \mathbf{x}_0, \mathbf{y}_0 \rangle]_T$ if and only if $S_k \subseteq S_c$.

The high-level idea of the construction of slotted uIPFE is similar to the uIPFE scheme in [TT18]. For ease of exposition, let us ignore the second slot of uIPFE for now. Informally, slot encryption for $(S_c, \{[\mathbf{x}_i]_1\}_{i \in S_c})$ chooses $z \leftarrow \mathbb{Z}_p$

[4] We require only indistinguishability-based security for unbounded slotted IPFE to prove simulation-based security of unbounded quadratic FE schemes. Note that the slotted property with indistinguishability-based security basically implies simulation-based security, and thus our approach essentially follows previous quadratic FE schemes with simulation-based security [Gay20, Wee20, GQ20].

[5] The second property is required for our unbounded quadratic FE from MDDH_k for $k > 1$ and FE for ABP $\circ$ UQF.

and encrypts $(\mathbf{u}_i, [\widetilde{\mathbf{x}}_i]_1)$ by slot encryption of pIPFE for all $i \in S_c$, where $\mathbf{u}_i = (1, i)$ and $\widetilde{\mathbf{x}}_i = (\mathbf{x}_i, z)$. Key generation for $(S_k, \{[\mathbf{y}_i]_2\}_{i \in S_k})$ chooses $a_i \leftarrow \mathbb{Z}_p$ so that $\sum_{i \in S_k} a_i = 0$ and computes a secret key of pIPFE for $(\mathbf{v}_i, [\widetilde{\mathbf{y}}_i]_1)$ for all $i \in S_k$, where $\mathbf{v}_i = (i, -1)$ and $\widetilde{\mathbf{y}}_i = (\mathbf{y}_i, a_i)$. Then, a decrypter can learn only $[\sum_{i \in S_c \cap S_k} \langle \mathbf{x}_i, \mathbf{y}_i \rangle + za_i]_T$ via decryption of pIPFE, where $za_i = 0$ only when $S_k \subseteq S_c$, and za_i looks random otherwise. Thus, we can recover $[\sum_{i \in S_k} \langle \mathbf{x}_i, \mathbf{y}_i \rangle]_T$ iff $S_k \subseteq S_c$. We defer how to obtain the slotted property to Section 4.

Put it all together. Let $\mathsf{uFE} = (\mathsf{uSetup}, \mathsf{uSlotEnc}, \mathsf{uEnc}, \mathsf{uKeyGen}, \mathsf{uDec})$ be a slotted uIPFE scheme and $H : \{0,1\}^* \to G_2$ be a hash function. Then, our unbounded quadratic FE scheme qFE is informally given as follows:

```
$\mathsf{Setup}(1^\lambda)$: $(\mathsf{PK}, \mathsf{MSK}) = (\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uSetup}(1^\lambda)$
$\mathsf{Enc}(\mathsf{PK}, (S_c, \{x_i\}_{i \in S_c}))$: $s_i \leftarrow \mathbb{Z}_p$, $\mathsf{uCT} \leftarrow \mathsf{uSlotEnc}(\mathsf{uPK}, (S_c, \{s_i\}_{i \in S_c}))$
    $\mathsf{iMSK} \leftarrow \mathsf{iSetup}(1^\lambda)$, $[w_i]_2 = H(i)$
    $\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [(x_i, s_i)]_1)$, $\mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [(x_i, w_i)]_2)$.
    $\mathsf{CT} = (\mathsf{uCT}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in S_c})$.
$\mathsf{KeyGen}(\mathsf{PK}, \mathsf{MSK}, (S_k, \{c_{i,j}\}_{i,j \in S_k}))$: $[w_i]_2 = H(i)$
    $\mathsf{SK} = \mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, \{[\sum_{j \in S_k} c_{i,j} w_j]_2\}_{i \in S_k}))$.
$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$: $\sum_{i,j \in S_k} c_{i,j} \mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j) - \mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK}) = [\sum_{i,j \in S_k} c_{i,j} x_i x_j]_T$.
```

Since the ciphertext size of slotted uIPFE is linear in $|S_c|$, that of the above quadratic FE scheme is also linear in $|S_c|$. The variable-length scheme without random oracles can be obtained by generating $[w_1]_2, \ldots, [w_{n'}]_2$ in the setup.

Security. Simulation-based security essentially asserts that a challenge ciphertext can be simulated without a challenge message, and secret keys can be simulated from corresponding decryption values. In our scheme, the simulation algorithms leverage the second slot of slotted uIPFE scheme uFE. Specifically, a simulated ciphertext is generated in the same manner as Enc except that uCT additionally encrypts $[1]_1$ (the generator of g_1) in the second slot, and $\mathsf{iCT}_i, \mathsf{iSK}_i$ encrypt $[(0, s_i)]_1, [(0, w_i)]_2$ instead of $[(x_i, s_i)]_1, [(x_i, w_i)]_2$, respectively. A simulated secret key for decryption value α is a secret key uSK of uFE that additional encodes $[-\alpha]_2$ if $S_k \subseteq S_c$ and $[0]_2$ otherwise in the second slot. Thanks to the slotted property, $-\mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK}) = [-\sum_{i,j \in S_k} c_{i,j} s_i w_j + \alpha]_T$ if $S_k \subseteq S_c$ in the above setting and the simulation goes well.

The indistinguishability between the real system and the simulated system can be proven by a series of hybrids similar to that used in Lin's secret-key quadratic FE scheme. Concretely, in the ℓ-th hybrid for $\ell \in S_c$, iCT_i and iSK_i is encrypting vectors $\mathbf{x}_i$ and $\widetilde{\mathbf{x}}_i$ where

$$\mathbf{x}_i = \begin{cases} (0,\ s_i) & (i \leq \ell) \\ (x_i, s_i) & (i > \ell) \end{cases}, \quad \widetilde{\mathbf{x}}_i = (x_i, w_i)$$

However, this change is detectable by decrypting the challenge ciphertext, and we need to adjust the difference using the second slot of uFE in each hybrid. Concretely, we encode $[1]_1$ into the second slot of uCT in the challenge ciphertext and $[-\sum_{i \in S_c^\ell \cap S_k, j \in S_k} c_{i,j} x_i x_j]_2$ into the second slot of uSK iff $S_k \subseteq S_c$, where S_c^ℓ

denotes the set consisting of the first ℓ elements of S_c. The indistinguishability between the $\ell-1$-th hybrid and the ℓ-th hybrid can be proven similarly to the proof of Lin's scheme. Observe that, in the final hybrid, the view of the adversary basically corresponds to that in the simulated system.

Extension to FE for ABP $\circ$ UQF. The high-level idea to extend our unbounded quadratic FE to FE for ABP $\circ$ UQF is similar to the technique used when achieving unboundedness in quadratic FE. That is, we can basically obtain FE for ABP $\circ$ UQF by enhancing the unbounded uIPFE uFE so that it can compute ABPs on a public input and linear functions on a private input. A similar idea is also used in the construction of Wee's recent partially-hiding FE scheme [Wee20]. We use a partially garbling scheme (PGS) for ABPs [IW14] for a building block.

We can formulate PGS for ABPs as follows. A garbling algorithm pgb takes an ABP $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$, a public input $\mathbf{u} \in \mathbb{Z}_p^n$, a private input $\mathbf{x} \in \mathbb{Z}_p^{n'}$, a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$ and outputs

$$\boldsymbol{\ell} = (\mathbf{u}'^\top \mathbf{L}_1 \mathbf{t}, \ldots, \mathbf{u}'^\top \mathbf{L}_m \mathbf{t}, x_1 + \mathbf{u}'^\top \mathbf{L}_{m+1} \mathbf{t}, \ldots, x_{n'} + \mathbf{u}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $\mathbf{u}' = (\mathbf{u}, 1)$, the parameter t and matrices $\mathbf{L}_i \in \mathbb{Z}_p^{(n+1)\times(t-1)}$ are determined by f, and $m = t - n'$. The correctness of the PGS requires that we can reconstruct $\langle f(\mathbf{u}), \mathbf{x}\rangle$ given $\boldsymbol{\ell}$ together with f and $\mathbf{u}$. Furthermore, the reconstruction is linear in $\boldsymbol{\ell}$, that is, there exists $\mathbf{d}_{f,\mathbf{u}} \in \mathbb{Z}_p^t$ and we have $\langle \mathbf{d}_{f,\mathbf{u}}, \boldsymbol{\ell}\rangle = \langle f(\mathbf{u}), \mathbf{x}\rangle$. The PGS is secure if there is an efficient algorithm pgb^* that takes $(f, \mathbf{u}, \alpha, \mathbf{t})$ for $\alpha \in \mathbb{Z}_p$, and the output distributions of $\mathsf{pgb}(f, \mathbf{u}, \mathbf{x}; \mathbf{t})$ and $\mathsf{pgb}^*(f, \mathbf{u}, \langle f(\mathbf{u}), \mathbf{x}\rangle; \mathbf{t})$ are statistically close where the probability is taken over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$.

Given the PGS for ABPs, we modify our unbounded quadratic FE scheme qFE to obtain FE for ABP $\circ$ UQF as follows. In encryption of $(\mathbf{u}, S_c, \{x_i\}_{i\in S_c})$, now uCT encrypts $r\mathbf{u}'$ with respect to identifier p in addition to $\{s_i\}_{i\in S_c}$ where $r \leftarrow \mathbb{Z}_p$ (recall that $S_c \subseteq [p-1]$ in FE for ABP $\circ$ UQF). A secret key for (S_k, f) consists of a set $\{\mathsf{uSK}_h\}_{h\in[t]}$ of secret keys of slotted uIPFE where uSK_h encodes $[w_i]_2$ for $i \in S_k$ and $\mathbf{L}_h\mathbf{t}$ such that $\mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK}_h)$ decrypts to the h-th element of $[\boldsymbol{\ell}]_T$ where

$$\boldsymbol{\ell} = (r\mathbf{u}'^\top \mathbf{L}_1 \mathbf{t}, \ldots, r\mathbf{u}'^\top \mathbf{L}_m \mathbf{t}, (s_i w_j + r\mathbf{u}'^\top \mathbf{L}_{\phi(i,j)}\mathbf{t})_{i,j\in S_k}) \in \mathbb{Z}_p^t \tag{1.1}$$

and $\phi : S_k \times S_k \to \{m+1, \ldots, t\}$ is a bijective function. Then, the decryption works as follows:

$$\begin{aligned}&\sum_{i,j\in S_k} f_{i,j}(\mathbf{u})\mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j) - \langle \mathbf{d}_{f,\mathbf{u}}, [\boldsymbol{\ell}]_T\rangle \\ =&[\sum_{i,j\in S_k} f_{i,j}(\mathbf{u})(x_i x_j + s_i w_j)]_T - [\sum_{i,j\in S_k} f_{i,j}(\mathbf{u}) s_i w_j]_T = [\sum_{i,j\in S_k} f_{i,j}(\mathbf{u}) x_i x_j]_T\end{aligned}$$

where the first equality follows from the correctness of the PGS.

The simulation algorithms of this extension scheme can be constructed in a similar manner to our unbounded quadratic FE scheme. A simulated ciphertext is the same as a normal ciphertext except that uCT encrypts $[\mathbf{0}]_1$ in the first slot and $[1]_1$ in the second slot, and $\mathsf{iCT}_i, \mathsf{iSK}_i$ encrypt $[(0, s_i)]_1, [(0, w_i)]_2$ instead of $[(x_i, s_i)]_1, [(x_i, w_i)]_2$, respectively. A simulated secret key for decryption value α is the same as a normal secret key except that the h-th element of $[\mathsf{pgb}^*(f, \mathbf{u}, -\alpha + \sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) s_i w_j, \mathbf{t})]_2$ (if $S_k \subseteq S_c$) or 0 (if $S_k \not\subseteq S_c$) is encoded in the second slot of uSK_h, where $\mathbf{t}$ is a random vector in $\mathbb{Z}_p^{t-1}$. In this simulation, we have $\langle \mathbf{d}_{f,\mathbf{u}}, (\mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK}_h)_{h \in [t]} \rangle = [-\alpha + \sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) s_i w_j]_T$ if $S_k \subseteq S_c$, and thus the simulation works.[6]

The intuition for the indistinguishability between the real system and the simulated system is given as follows. The adversary in the real system can basically learn $\{[x_i x_j + s_i w_j]_T\}_{i,j \in S_c}$ from $\mathsf{iCT}_i, \mathsf{iSK}_i$ in the challenge ciphertext and $[\boldsymbol{\ell}]_T$ defined in Eq. (1.1) with respect to secret keys for $S_k \subseteq S_c$ from $\mathsf{uCT}, \mathsf{uSK}_h$. Under the SXDH, the adversary cannot detect the change if $\boldsymbol{\ell}$ is computed as

$$\boldsymbol{\ell} = (\mathbf{u}'^{\top} \mathbf{L}_1 \widetilde{\mathbf{t}}, \ldots, \mathbf{u}'^{\top} \mathbf{L}_m \widetilde{\mathbf{t}}, (s_i w_j + \mathbf{u}'^{\top} \mathbf{L}_{\phi(i,j)} \widetilde{\mathbf{t}})_{i,j \in S_k})$$

where $\widetilde{\mathbf{t}}$ is a random vector that is independent of $\mathbf{t}$ used in generating uSK_h. Then, due to the security of the PGS, $\boldsymbol{\ell}$ reveals only $[\sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) s_i w_j]_T$. Again, $\{s_i w_j\}_{i,j \in S_c}$ looks random under the SXDH, and thus we have

$$\begin{aligned} &\{\{[x_i x_j + s_i w_j]_T\}_{i,j \in S_c}, [\sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) s_i w_j]_T\} \\ \approx_c\ &\{\{[s_i w_j]_T\}_{i,j \in S_c}, [\sum_{i,j \in S_k} f_{i,j}(\mathbf{u})(s_i w_j - x_i x_j)]_T\} \end{aligned}$$

where the RHS basically corresponds to the view in the simulated system.

2 Preliminaries

2.1 Notations

For $m \in \mathbb{N}$, $[m]$ denotes a set $\{1, \ldots, m\}$. For vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$, $(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ denotes the vector concatenation as row vectors *regardless of* whether each $\mathbf{v}_i$ is a row or column vector. For instance, for $\mathbf{v}_1 \in \mathbb{Z}_p^{m \times 1}, \mathbf{v}_2 \in \mathbb{Z}_p^{1 \times n}$, $(\mathbf{v}_1, \mathbf{v}_2) = (\mathbf{v}_1^{\top} || \mathbf{v}_2)$. 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. We use $\otimes$ for the Kronecker product. For a matrix $\mathbf{M} \in \mathbb{Z}_p^{a \times b}$ and vectors $\mathbf{a} \in \mathbb{Z}_p^a, \mathbf{b} \in \mathbb{Z}_p^b$, we denote a vector $\mathbf{m}$ such that $\langle \mathbf{a} \otimes \mathbf{b}, \mathbf{m} \rangle = \mathbf{a}^{\top} \mathbf{M} \mathbf{b}$ by $\mathsf{vec}(\mathbf{M})$. For families of distributions $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$, we denote $X \approx_c Y$ and $X \approx_s Y$ as computational indistinguishability and statistical indistinguishability, respectively.

[6] It is not hard to see that the security of the partially garbling scheme implies that $\langle \mathbf{d}_{f,\mathbf{u}}, \mathsf{pgb}^*(f, \mathbf{u}, \alpha; \mathbf{t}) \rangle = \alpha$ for all $\alpha \in \mathbb{Z}_p$.

2.2 Basic Tools and Assumptions

Definition 2.1 (Bilinear Groups). Let $\{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ be a family of bilinear groups. Bilinear groups $\mathbb{G}_\lambda=(p, G_1, G_2, G_T, g_1, g_2, e)$ are specified by a prime p, cyclic groups G_1, G_2, G_T of order p, generators g_1 and g_2 of G_1 and G_2 respectively, and a bilinear map $e : G_1 \times G_2 \to G_T$, which has two properties.

– (Bilinearity): $\forall h_1 \in G_1, h_2 \in G_2, a, b \in \mathbb{Z}_p, e(h_1^a, h_2^b) = e(h_1, h_2)^{ab}$.
– (Non-degeneracy): For g_1 and g_2, $g_T = e(g_1, g_2)$ is a generator of G_T.

In what follows, we omit the index λ from $\mathbb{G}_\lambda$ and abuse notation by denoting a family of bilinear groups $\{\mathbb{G}_\lambda\}_{\lambda\in\mathbb{N}}$ also by $\mathbb{G}$ if it is clear in the context.

Definition 2.2 ($\mathcal{D}_{j,k}$-MDDH Assumption [EHK⁺17]). Let $\{\mathbb{G}\}$ be a family of bilinear groups. For $j > k$, let $\mathcal{D}_{j,k}$ be a matrix distribution over matrices in $\mathbb{Z}_p^{j\times k}$, which outputs a full-rank matrix with overwhelming probability. We can assume that, wlog, the first k rows of a matrix chosen from $\mathcal{D}_{j,k}$ form an invertible matrix. We consider the following distribution: $\mathbf{A} \leftarrow \mathcal{D}_{j,k},\ \mathbf{z} \leftarrow \mathbb{Z}_p^k,\ \mathbf{k}_0 = \mathbf{A}\mathbf{z},\ \mathbf{k}_1 \leftarrow \mathbb{Z}_p^j,\ P_{i,\beta} = (\mathbb{G}, [\mathbf{A}]_i, [\mathbf{k}_\beta]_i)$. We say that the $\mathcal{D}_{j,k}$-MDDH assumption holds with respect to $\{\mathbb{G}\}$ if, for any PPT adversary $\mathcal{A}$,

$$\mathsf{Adv}_{\mathcal{A}}^{\mathcal{D}_{j,k}\text{-}\mathsf{MDDH}} = \max_{i\in\{1,2\}} |\Pr[1 \leftarrow \mathcal{A}(P_{i,0})] - \Pr[1 \leftarrow \mathcal{A}(P_{i,1})]| \le \mathsf{negl}(\lambda).$$

In what follows, we denote $\mathcal{D}_{k+1,k}$ by $\mathcal{D}_k$. Note that the well-known k-Lin assumption can be captured as the $\mathcal{D}_k$-MDDH assumption.

Uniform Distribution. Let $\mathcal{U}_{j,k}$ be a uniform distribution over $\mathbb{Z}_p^{j\times k}$. Then, the following holds with tight reductions: $\mathcal{D}_k$-MDDH $\Rightarrow$ $\mathcal{U}_k$-MDDH $\Rightarrow$ $\mathcal{U}_{j,k}$-MDDH. We denote $\mathcal{D}_k$-MDDH by MDDH$_k$.

Definition 2.3 (Arithmetic Branching Programs (ABPs)). An arithmetic branching program $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$ is defined by a prime p, a directed acyclic graph (V, E), two special vertices $v_0, v_1 \in V$, and a labeling function $\sigma : E \to \mathcal{F}^{\mathsf{Affine}}$, where $\mathcal{F}^{\mathsf{Affine}}$ consists of all affine functions $g : \mathbb{Z}_p^n \to \mathbb{Z}_p$. The size of f is the number of vertices $|V|$. Given an input $\mathbf{x} \in \mathbb{Z}_p^n$ to the ABP, we can assign a $\mathbb{Z}_p$ element to edge $e \in E$ by $\sigma(e)(\mathbf{x})$. Let P be the set of all paths from v_0 to v_1. Each element in P can be represented by a subset of E. The output of the ABP on input $\mathbf{x}$ is defined as $\sum_{E'\in P} \prod_{e\in E'} \sigma(e)(\mathbf{x})$. We can extend the definition of ABPs for functions $f : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n'}$ by evaluating each output in a coordinate-wise manner and denote such a function class by $\mathcal{F}_{n,n'}^{\mathsf{ABP}}$.

Note that we can convert any boolean formula, boolean branching program or arithmetic formula to an arithmetic branching program with a constant blow-up in the representation size. Thus, ABPs are a stronger computational model than all of the above.

2.3 Functional Encryption

We first define functional encryption (FE). In FE, the system can generate a secret key that is associated with a function f, and a ciphertext for a message x decrypts to $f(x)$ when it is decrypted by the secret key for f. Typically, FE is defined as the ciphertext for x entirely hides x. Recently, the more generalized notion called partially hiding FE [AJS18] was introduced, where the ciphertext of x partially hides x. More precisely, x consists of the public part x_{pub} and the private part x_{priv}, and the ciphertext for x hides only x_{priv}. In this paper, we use several classes of partially hiding FE, which is formally defined as follows.

Definition 2.4 (Functional Encryption). Let $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}}$ be a message space. Let $\mathcal{F}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X} \to \mathcal{Z}$. A (public-key) functional encryption (FE) scheme for $\mathcal{F}$, FE, consists of four algorithms.

$\mathsf{Setup}(1^\lambda)$**:** It takes a security parameter 1^λ and outputs a public parameter PK and a master secret key MSK. The other three algorithms implicitly take PK as input.

$\mathsf{Enc}(x)$**:** It takes $x \in \mathcal{X}$ and outputs a ciphertext CT.

$\mathsf{KeyGen}(\mathsf{MSK}, f)$**:** It takes MSK and $f \in \mathcal{F}$, and outputs a secret key SK.

$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** It takes CT and SK, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. FE is *correct* if it satisfies the following condition. For all $\lambda \in \mathbb{N}$, $x \in \mathcal{X}$, $f \in \mathcal{F}$, we have

$$\Pr\left[\mathsf{Dec}(\mathsf{CT},\mathsf{SK}) = f(x) \;\middle|\; \begin{array}{l} \mathsf{PK},\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(x) \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \end{array}\right] = 1.$$

Security. We define two types of partially-hiding security for FE, namely, indistinguishability-based security and simulation-based security[7]. We first define indistinguishability-based security. For a stateful PPT adversary $\mathcal{A}$ and $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathsf{i\text{-}ph}}(\lambda) = \left| \Pr\left[\beta' = \beta \;\middle|\; \begin{array}{l} \beta \leftarrow \{0,1\}, \;\; \mathsf{PK},\mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda) \\ (x_{\mathsf{pub}}, x^0_{\mathsf{priv}}, x^1_{\mathsf{priv}}) \leftarrow \mathcal{A}(\mathsf{PK}) \\ \mathsf{CT} \leftarrow \mathsf{Enc}((x_{\mathsf{pub}}, x^\beta_{\mathsf{priv}})) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{CT}) \end{array}\right] - 1/2 \right| \tag{2.1}$$

Let q_k be a number of queries to KeyGen and f^ℓ be the ℓ-th function on which $\mathcal{A}$ queries KeyGen. We say $\mathcal{A}$ is *admissible* if $\mathcal{A}$'s queries satisfy the followings:

$$f^\ell((x_{\mathsf{pub}}, x^0_{\mathsf{priv}})) = f^\ell((x_{\mathsf{pub}}, x^1_{\mathsf{priv}})) \;\text{ for all } \ell \in [q_k].$$

[7] We consider only selective (or semi-adaptive more precisely) security in this paper. It is well-known that semi-adaptive simulation-based security implies semi-adaptive indistinguishability-based security.

FE is said to be *IND-partially hiding* if, for all admissible PPT adversaries $\mathcal{A}$, we have $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathsf{i\text{-}ph}}(\lambda) \leq \mathsf{negl}(\lambda)$.

Next we define simulation-based security. For a stateful PPT adversary $\mathcal{A}$, $\lambda \in \mathbb{N}$, and simulation algorithms $(\mathsf{Setup}^*, \mathsf{Enc}^*, \mathsf{KeyGen}^*)$ let

$$\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathsf{s\text{-}ph}}(\lambda) = \left| \begin{array}{l} \Pr\left[\beta = 1 \;\middle|\; \begin{array}{l} \mathsf{PK}, \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda) \\ x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \leftarrow \mathcal{A}(\mathsf{PK}) \\ \mathsf{CT} \leftarrow \mathsf{Enc}((x_{\mathsf{pub}}, x_{\mathsf{priv}})) \\ \beta \leftarrow \mathcal{A}^{\mathsf{KeyGen}(\mathsf{MSK},\cdot)}(\mathsf{CT}) \end{array} \right] \\ -\Pr\left[\beta = 1 \;\middle|\; \begin{array}{l} \mathsf{PK}^*, \mathsf{MSK}^* \leftarrow \mathsf{Setup}^*(1^\lambda) \\ x = (x_{\mathsf{pub}}, x_{\mathsf{priv}}) \leftarrow \mathcal{A}(\mathsf{PK}^*) \\ \mathsf{CT}^* \leftarrow \mathsf{Enc}^*(\mathsf{MSK}^*, x_{\mathsf{pub}}) \\ \beta \leftarrow \mathcal{A}^{\mathsf{KeyGen}^*(\mathsf{MSK}^*,\cdot)}(\mathsf{CT}^*) \end{array} \right] \end{array} \right| \tag{2.2}$$

where KeyGen^* obtains $f(x)$ and x_{pub} together with f whenever $\mathcal{A}$ queries f to KeyGen^*. FE is said to be *SIM-partially hiding* if, for all $\mathcal{A}$, there exist simulation algorithms $(\mathsf{Setup}^*, \mathsf{Enc}^*, \mathsf{KeyGen}^*)$ and we have $\mathsf{Adv}^{\mathsf{FE}}_{\mathcal{A},\mathsf{s\text{-}ph}}(\lambda) \leq \mathsf{negl}(\lambda)$. It is not well-known that simulation-based security implies indistinguishability-based security in (public-key) FE.

Next, we define a more generalized notion that we call slotted functional encryption. Slotted FE was first introduced in [LV16] for inner product functionality, which is called slotted inner product FE. We extend it to handle general functions since we use slotted FE schemes for other classes in this paper.

Before explaining the definition of slotted FE, let us recall the notion of function-hiding FE. In function-hiding FE, a secret key for f hides f as well as a ciphertext for x hides x. We usually consider the secret-key setting where encryption requires a master secret key for function-hiding FE. This is because an adversary can learn $f(x)$ for any x from a secret key for f in public-key FE, and it is difficult to achieve meaningful function-hiding security.

Slotted FE is a hybrid between public-key FE and function-hiding secret-key FE. In slotted FE, a private message space $\mathcal{X}_{\mathsf{priv}}$ consists of two spaces $\mathcal{X}_{\mathsf{priv1}}$ and $\mathcal{X}_{\mathsf{priv2}}$, that is, a massage space consists of three spaces: $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}} \times \mathcal{X}_{\mathsf{priv2}}$. For some default value $e \in \mathcal{X}_{\mathsf{priv2}}$, a user can publicly encrypt $(x_{\mathsf{pub}}, x_{\mathsf{priv}}, e) \in \mathcal{X}$ for all $(x_{\mathsf{pub}}, x_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}}$ while an owner of master secret key can encrypt all $x \in \mathcal{X}$. On the other hand, a function space $\mathcal{F}$ consists of two spaces $\mathcal{F}_{\mathsf{pub}}$ and $\mathcal{F}_{\mathsf{priv}}$. A secret key for $f = (f_{\mathsf{pub}}, f_{\mathsf{priv}}) \in \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ hides f_{priv}. Intuitively, meaningful function-hiding security with respective to $\mathcal{F}_{\mathsf{priv}}$ can be achieved by the fact that the adversary can encrypt only messages of the form $(x_{\mathsf{pub}}, x_{\mathsf{priv}}, e) \in \mathcal{X}$. Slotted FE is formally defined as follows.

Definition 2.5 (Slotted Functional Encryption). Let $\mathcal{X} = \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}} \times \mathcal{X}_{\mathsf{priv2}}$ be a message space. We sometimes denote $\mathcal{X}_{\mathsf{priv1}} \times \mathcal{X}_{\mathsf{priv2}}$ by $\mathcal{X}_{\mathsf{priv}}$. Let $\mathcal{F} = \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ be a function family such that, for all $f \in \mathcal{F}$, $f : \mathcal{X} \to \mathcal{Z}$. A slotted functional encryption (SlotFE) scheme for $\mathcal{F}$, SlotFE, consists of five algorithms.

$\mathsf{Setup}(1^\lambda)$**:** It takes a security parameter 1^λ and outputs a public key PK and a master secret key MSK. The other four algorithms implicitly take PK as input.
$\mathsf{Enc}(\mathsf{MSK}, x)$**:** It takes MSK and $x \in \mathcal{X}$ and outputs a ciphertext CT.
$\mathsf{SlotEnc}(x)$**:** It takes $x \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}}$ and outputs a ciphertext CT.
$\mathsf{KeyGen}(\mathsf{MSK}, f)$**:** It takes MSK and $f \in \mathcal{F}$, and outputs a secret key SK.
$\mathsf{Dec}(\mathsf{CT}, \mathsf{SK})$**:** It takes CT and SK, and outputs a decryption value $d \in \mathcal{Z}$ or a symbol $\perp$.

Correctness. SlotFE is *correct* if it satisfies the following condition. For all $\lambda \in \mathbb{N},\ x \in \mathcal{X},\ f \in \mathcal{F}$, we have

$$\Pr\left[\mathsf{Dec}(\mathsf{CT}, \mathsf{SK}) = f(x) \;\middle|\; \begin{array}{l} \mathsf{PP}, \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda) \\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, x) \\ \mathsf{SK} \leftarrow \mathsf{KeyGen}(\mathsf{MSK}, f) \end{array}\right] = 1.$$

Slot-mode correctness. SlotFE is *slot-mode correct* with respect to a public element $e \in \mathcal{X}_{\mathsf{priv2}}$ if it satisfies the following condition. For all $\lambda \in \mathbb{N},\ x \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}}$, the following distributions are identical:

$$\left\{(\mathsf{PK}, \mathsf{MSK}, \mathsf{CT}) \mid (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda),\ \mathsf{CT} \leftarrow \mathsf{Enc}(\mathsf{MSK}, (x, e))\right\}$$
$$\left\{(\mathsf{PK}, \mathsf{MSK}, \mathsf{CT}) \mid (\mathsf{PK}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda),\ \mathsf{CT} \leftarrow \mathsf{SlotEnc}(x)\right\}$$

Security. We define partially-hiding security for SlotFE. For slotted FE, we consider only indistinguishability-based security. For a stateful PPT adversary $\mathcal{A}$ and $\lambda \in \mathbb{N}$, let

$$\mathsf{Adv}^{\mathsf{SlotFE}}_{\mathcal{A},\mathsf{ph}} = \left| \Pr\left[\beta' = \beta \;\middle|\; \begin{array}{l} \beta \leftarrow \{0,1\},\ \ \mathsf{PK}, \mathsf{MSK} \leftarrow \mathsf{Setup}(1^\lambda) \\ \beta' \leftarrow \mathcal{A}^{\mathsf{cO}(\beta,\cdot), \mathsf{kO}(\beta,\cdot)}(\mathsf{PK}) \end{array}\right] - 1/2 \right| \tag{2.3}$$

where $\mathsf{cO}(\beta, \cdot)$ takes $(x_{\mathsf{pub}}, x^0_{\mathsf{priv}}, x^1_{\mathsf{priv}}) \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}^2_{\mathsf{priv}}$ and returns $\mathsf{Enc}(\mathsf{MSK}, (x_{\mathsf{pub}}, x^\beta_{\mathsf{priv}}))$, $\mathsf{kO}(\beta, \cdot)$ takes $(f_{\mathsf{pub}}, f^0_{\mathsf{priv}}, f^1_{\mathsf{priv}}) \in \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}^2_{\mathsf{priv}}$ and returns $\mathsf{KeyGen}(\mathsf{MSK}, (f_{\mathsf{pub}}, f^\beta_{\mathsf{priv}}))$. Let q_c, q_k be a number of queries to cO, kO, respectively. Let $x^{j,\beta} = (x^j_{\mathsf{pub}}, x^{j,\beta}_{\mathsf{priv}})$ for $j \in [q_c]$, and $f^{\ell,\beta} = (f^\ell_{\mathsf{pub}}, f^{\ell,\beta}_{\mathsf{priv}})$ for $\ell \in [q_k]$. We say $\mathcal{A}$ is *admissible* if $\mathcal{A}$'s queries satisfy the followings:

– $\mathcal{A}$ never queries cO after querying kO even once[8];
– $f^{\ell,0}(x^{j,0}) = f^{\ell,1}(x^{j,1})$ for all $j \in [q_c], \ell \in [q_k]$; and
– $f^{\ell,0}((x, e)) = f^{\ell,1}((x, e))$ for all $\ell \in [q_k], x \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}}$ where e is the public element defined in slot-mode correctness[9].

[8] This condition implies selective security (or semi-adaptive security more precisely).
[9] In general, this condition is necessary since the adversary can publicly encrypt (x, e) for all $x \in \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}}$ and decrypt the ciphertexts with its own secret keys. In this paper, however, we handle only function classes where this condition is always satisfied as long as the public parts of $f^{\ell,0}$ and $f^{\ell,1}$ are the same. Thus, we can ignore this condition in this paper.

SlotFE is said to be *IND-q_c-partially hiding* if, for all admissible PPT adversaries $\mathcal{A}$ querying cO at most q_c times, we have $\mathsf{Adv}^{\mathsf{SlotFE}}_{\mathcal{A},\mathsf{ph}} \leq \mathsf{negl}(\lambda)$. When q_c can be any polynomial, i.e., $q_c = \mathsf{poly}(\lambda)$, we call the scheme just *IND-partially hiding*.

We define slotted FE for inner product over bilinear groups called slotted IPFE, which we extensively use in this paper. A concrete construction of slotted IPFE is found in [LL20, Appendix A].

Definition 2.6 (Slotted IPFE). Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups, $\mathcal{X}_{\mathsf{pub}} = \emptyset, \mathcal{X}_{\mathsf{priv1}} = G_1^{m_1}, \mathcal{X}_{\mathsf{priv2}} = G_1^{m_2}, \mathcal{F}_{\mathsf{pub}} = G_2^{m_1}, \mathcal{F}_{\mathsf{priv}} = G_2^{m_2}$. A function family $\mathcal{F}^{\mathsf{IP}}_{m_1,m_2,\mathbb{G}} = \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ consists of functions $f : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}} \times \mathcal{X}_{\mathsf{priv2}} \to G_T \cup \{\perp\}$. Each $f \in \mathcal{F}^{\mathsf{IP}}_{m_1,m_2,\mathbb{G}}$ is specified by $([\mathbf{y}_1]_2, [\mathbf{y}_2]_2) \in \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ where $\mathbf{y}_i = \mathbb{Z}_p^{m_i}$ and defined as

$$f([\mathbf{x}_1]_1, [\mathbf{x}_2]_1) = [\langle \mathbf{x}_1, \mathbf{y}_1 \rangle + \langle \mathbf{x}_2, \mathbf{y}_2 \rangle]_T$$

where $\mathbf{x}_i \in \mathbb{Z}_p^{m_i}$. We refer to slotted FE for $\mathcal{F}^{\mathsf{IP}}_{m_1,m_2,\mathbb{G}}$ as slotted IPFE. Note that when $m_1 = 0$, slotted IPFE corresponds to secret-key function-hiding IPFE.

We define FE for unbounded quadratic functions (UQF) and its extension to the combination with ABPs (ABP$\circ$UQF). Our goal in this paper is to construct FE (not slotted FE) schemes for the two functionalities. We formally define the two functionalities as follows.

Definition 2.7 (Unbounded Quadratic Functional Encryption). Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups, $\mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} = \{(x_1, x_2) \in 2^{[p]} \times \bigcup_{i\in[p]} \mathbb{Z}_p^i \mid |x_1| = |x_2|\}$ where $|x_1|$ denotes the cardinality of x_1, and $|x_2|$ denotes the length of x_2. Let $\mathcal{F} = \{(f_1, f_2) \in 2^{[p]} \times \bigcup_{i\in[p]} \mathbb{Z}_p^{i^2} \mid |f_1|^2 = |f_2|\}$. A function family $\mathcal{F}^{\mathsf{UQF}}_{\mathbb{G}} = \mathcal{F}$ consists of functions $f : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to G_T \cup \{\perp\}$. Each $f \in \mathcal{F}^{\mathsf{UQF}}_{\mathbb{G}}$ is specified by $(S_k, \mathbf{c}) \in \mathcal{F}$ where $S_k \subseteq [p], \mathbf{c} = (c_{i.j})_{i.j \in S_k} \in (\mathbb{Z}_p)^{S_k \times S_k}$ and defined as

$$f((S_c, \mathbf{x})) = \begin{cases} [\sum_{i,j\in S_k} c_{i,j} x_i x_j]_T & S_k \subseteq S_c \\ \perp & \text{otherwise} \end{cases}$$

where $S_c \subseteq [p], \mathbf{x} = (x_i)_{i\in S_c} \in \mathbb{Z}_p^{S_c}$. Note that S_c is the public input while $\mathbf{x}$ is a private input. We refer to FE for $\mathcal{F}^{\mathsf{UQF}}_{\mathbb{G}}$ with the ciphertext-size being linear in $|S_c|$ as unbounded quadratic functional encryption.

Definition 2.8 (Functional Encryption for ABP$\circ$UQF). Let $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$ be bilinear groups, $q = p - 1$, $\mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} = \{((x_1, x_2), x_3) \in (\mathbb{Z}_p^n \times 2^{[q]}) \times \bigcup_{i\in[q]} \mathbb{Z}_p^i \mid |x_2| = |x_3|\}$ where $|x_2|$ denotes the cardinality of x_2, and $|x_3|$ denotes the length of x_3. Let $\mathcal{F} = \{(f_1, f_2) \in 2^{[q]} \times \bigcup_{i\in[q]} \mathcal{F}^{\mathsf{ABP}}_{n,i^2} \mid |f_1|^2 = \mathsf{OutLen}(f_2)\}$ where $|f_1|$ denotes the cardinality of f_1, and $\mathsf{OutLen}(f_2)$ denotes the output length of f_2. A function family $\mathcal{F}^{\mathsf{ABP}\circ\mathsf{UQF}}_{n,\mathbb{G}} = \mathcal{F}$ consists of functions

$f : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to G_T \cup \{\perp\}$. Each $f \in \mathcal{F}^{\mathsf{ABP}\circ\mathsf{UQF}}_{n,\mathbb{G}}$ is specified by $(S_k, f^{\mathsf{ABP}}) \in \mathcal{F}$ where $S_k \subseteq [q], f^{\mathsf{ABP}} \in \mathcal{F}^{\mathsf{ABP}}_{n,|S_k|^2}$ and defined as

$$f((\mathbf{u}, S_c, \mathbf{x})) = \begin{cases} [\sum_{i,j \in S_k} f^{\mathsf{ABP}}_{i,j}(\mathbf{u}) x_i x_j]_T & S_k \subseteq S_c \\ \perp & \text{otherwise} \end{cases}$$

where $\mathbf{u} \in \mathbb{Z}_p^n, S_c \subseteq [q], \mathbf{x} = (x_i)_{i \in S_c} \in \mathbb{Z}_p^{S_c}$ and $f^{\mathsf{ABP}}_{i,j}(\mathbf{u})$ is the (i,j)-th element of $f^{\mathsf{ABP}}(\mathbf{u})$. Note that $\mathbf{u}, S_c$ are the public input while $\mathbf{x}$ is a private input. We refer to FE for $\mathcal{F}^{\mathsf{ABP}\circ\mathsf{UQF}}_{n,\mathbb{G}}$ with the ciphertext-size being linear in $|S_c|$ and $|\mathbf{u}|$ as FE for ABP $\circ$ UQF.

Note that our scheme computes function values as an exponent of a group element where the discrete log problem is hard. Thus, we require the exponent to be in a polynomial range if the decrypter needs to obtain the function value as a $\mathbb{Z}_p$ element. Note that this restriction is common in all previous FE schemes for inner product or quadratic functions based on cyclic groups.

3 Predicate Slotted Inner Product Functional Encryption

In this section, we define a new primitive called predicate slotted IPFE and show how to construct it. We use it as a building block of our unbounded slotted IPFE scheme that we present in Section 4.

3.1 Definitions

Definition 3.1 (Predicate Slotted IPFE). Let $\mathbb{G}$ be bilinear groups, $\mathcal{X}_{\mathsf{pub}} = \mathbb{Z}_p^d, \mathcal{X}_{\mathsf{priv1}} = G_1^{m_1}, \mathcal{X}_{\mathsf{priv2}} = G_1^{m_2}, \mathcal{F}_{\mathsf{pub}} = \mathbb{Z}_p^d \times G_2^{m_1}, \mathcal{F}_{\mathsf{priv}} = G_2^{m_2}$. A function family $\mathcal{F}^{\mathsf{PIP}}_{d,m_1,m_2,\mathbb{G}} = \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ consists of functions $f : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}} \times \mathcal{X}_{\mathsf{priv2}} \to G_T \cup \{\perp\}$. Each $f \in \mathcal{F}^{\mathsf{PIP}}_{d,m_1,m_2,\mathbb{G}}$ is specified by $((\mathbf{v}, [\mathbf{y}_1]_2), [\mathbf{y}_2]_2) \in \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ where $\mathbf{v} \in \mathbb{Z}_p^d, \mathbf{y}_i \in \mathbb{Z}_p^{m_i}$ and defined as

$$f(\mathbf{u}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1) = \begin{cases} [\langle \mathbf{x}_1, \mathbf{y}_1 \rangle + \langle \mathbf{x}_2, \mathbf{y}_2 \rangle]_T & \text{if } \langle \mathbf{u}, \mathbf{v} \rangle = 0 \\ \perp & \text{if } \langle \mathbf{u}, \mathbf{v} \rangle \neq 0 \end{cases}$$

where $\mathbf{u} \in \mathbb{Z}_p^d, \mathbf{x}_i \in \mathbb{Z}_p^{m_i}$. We refer to slotted FE (Definition 2.5) for $\mathcal{F}^{\mathsf{PIP}}_{d,m_1,m_2,\mathbb{G}}$ as predicate slotted IPFE.

3.2 Predicate Slotted IPFE from Slotted IPFE

We construct a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{PIP}}_{d,m_1,m_2,\mathbb{G}}$ from a partially hiding FE scheme for $\mathcal{F}^{\mathsf{IP}}_{kd+m_1,2m_2+1,\mathbb{G}}$ in a generic way. Note that k is a parameter for the MDDH_k assumption.

Construction. Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{IP}}_{kd+m_1,2m_2+1,\mathbb{G}}$ with slot-mode correctness for $e = [0^{2m_2+1}]_1$. Then, our partially hiding slotted FE scheme $\mathsf{pFE} = (\mathsf{pSetup}, \mathsf{pEnc}, \mathsf{pSlotEnc}, \mathsf{pKeyGen}, \mathsf{pDec})$ for $\mathcal{F}^{\mathsf{PIP}}_{d,m_1,m_2,\mathbb{G}}$ with slot-mode correctness for $e = [0^{m_2}]_1$ is constructed as follows.

$\mathsf{pSetup}(1^\lambda)$**:** It outputs $(\mathsf{pPK}, \mathsf{pMSK}) = (\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$.
$\mathsf{pEnc}(\mathsf{pMSK}, (\mathbf{u}, [\mathbf{x}_1]_1, [\mathbf{x}_2]_1))$**:** It outputs pCT as follows:

$$\mathbf{z} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{x}}_1 = (\mathbf{z} \otimes \mathbf{u}, \mathbf{x}_1) \in \mathbb{Z}_p^{kd+m_1},\ \widetilde{\mathbf{x}}_2 = (\mathbf{x}_2, 0^{m_2}, 0) \in \mathbb{Z}_p^{2m_2+1}$$
$$\mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, ([\widetilde{\mathbf{x}}_1]_1, [\widetilde{\mathbf{x}}_2]_1)),\ \mathsf{pCT} = (\mathbf{u}, \mathsf{iCT}).$$

$\mathsf{pSlotEnc}(\mathbf{u}, [\mathbf{x}_1]_1)$**:** It outputs pCT as follows:

$$\mathbf{z} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{x}}_1 = (\mathbf{z} \otimes \mathbf{u}, \mathbf{x}_1) \in \mathbb{Z}_p^{kd+m_1},\ \mathsf{iCT} \leftarrow \mathsf{iSlotEnc}([\widetilde{\mathbf{x}}_1]_1),\ \mathsf{pCT} = (\mathbf{u}, \mathsf{iCT}).$$

$\mathsf{pKeyGen}(\mathsf{pMSK}, (\mathbf{v}, [\mathbf{y}_1]_2, [\mathbf{y}_2]_2))$**:** It outputs pSK as follows:

$$\mathbf{a} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{y}}_1 = (\mathbf{a} \otimes \mathbf{v}, \mathbf{y}_1) \in \mathbb{Z}_p^{kd+m_1},\ \widetilde{\mathbf{y}}_2 = (\mathbf{y}_2, 0^{m_2}, 0) \in \mathbb{Z}_p^{2m_2+1}$$
$$\mathsf{iSK} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, ([\widetilde{\mathbf{y}}_1]_1, [\widetilde{\mathbf{y}}_2]_1)),\ \mathsf{pSK} = (\mathbf{v}, \mathsf{iSK}).$$

$\mathsf{pDec}(\mathsf{pCT}, \mathsf{pSK})$**:** If $\langle \mathbf{u}, \mathbf{v} \rangle \neq 0$, it outputs $\perp$. Otherwise, outputs $\mathsf{iDec}(\mathsf{iCT}, \mathsf{iSK})$.

Correctness. Since $\langle \mathbf{z} \otimes \mathbf{u}, \mathbf{a} \otimes \mathbf{v} \rangle = \langle \mathbf{z}, \mathbf{a} \rangle \cdot \langle \mathbf{u}, \mathbf{v} \rangle$, $\mathsf{iDec}(\mathsf{iCT}, \mathsf{iSK})$ outputs $[\langle \widetilde{\mathbf{x}}_1, \widetilde{\mathbf{y}}_1 \rangle + \langle \widetilde{\mathbf{x}}_2, \widetilde{\mathbf{y}}_2 \rangle]_T = [\langle \mathbf{x}_1, \mathbf{y}_1 \rangle + \langle \mathbf{x}_2, \mathbf{y}_2 \rangle]_T$ if $\langle \mathbf{u}, \mathbf{v} \rangle = 0$. This follows from the correctness of iFE.

Slot-mode correctness. Thanks to slot-mode correctness of iFE, $\mathsf{iSlotEnc}([\widetilde{\mathbf{x}}_1]_1)$ and $\mathsf{iEnc}(\mathsf{iMSK}, ([\widetilde{\mathbf{x}}_1]_1, [0^{2m_2+1}]_1))$ are identically distributed for all correctly generated $(\mathsf{iMSK}, \mathsf{iPK})$ and $\widetilde{\mathbf{x}}_1 \in \mathbb{Z}_p^{kd+m_1}$. Hence, $\mathsf{pSlotEnc}(\mathbf{u}, [\mathbf{x}_1]_1)$ and $\mathsf{pEnc}(\mathsf{pMSK}, (\mathbf{u}, [\mathbf{x}_1]_1, [0^{m_2}]_1))$ are identically distributed for all correctly generated $(\mathsf{pMSK}, \mathsf{pPK})$, $\mathbf{u} \in \mathbb{Z}_p^d$, and $\mathbf{x}_1 \in \mathbb{Z}_p^{m_1}$.

3.3 Security Analysis

For security, we have the following theorem.

Theorem 3.1. *If* iFE *is IND-partially hiding, and the MDDH*$_k$ *assumption holds in* $\mathbb{G}$*, then* pFE *is IND-partially hiding.*

Proof. We prove Theorem 3.1 via a series of hybrid games $\mathsf{H}^\beta_{\iota,1}, \ldots, \mathsf{H}^\beta_{\iota,4}$ for $\iota \in [q_c]$ and H^β_f. We show that $\mathsf{H}^\beta_s \approx_c \mathsf{H}^\beta_{1,1} \approx_c \cdots \approx_c \mathsf{H}^\beta_{1,4} \approx_c \mathsf{H}^\beta_{2,1} \approx_c \cdots \approx_c \mathsf{H}^\beta_{q_c,4} \approx_c \mathsf{H}^\beta_f$, where H^β_s for $\beta \in \{0,1\}$ is the original security game (described in Eq. (2.3)). Especially, the oracles cO and kO works as Fig 1 in H^β_s. In the hybrid sequence, the behavior of the oracles is gradually changed. Each hybrid is defined as follows.

$\mathsf{H}^\beta_{\iota,1}$**:** This game is the same as H^β_s except that

$\underline{\mathsf{cO}(\beta,\cdot)}$	$\underline{\mathsf{kO}(\beta,\cdot)}$
Input: $\mathbf{u} \in \mathcal{X}_{\mathsf{pub}}, ([(\mathbf{x}_1^0, \mathbf{x}_2^0)]_1, [(\mathbf{x}_1^1, \mathbf{x}_2^1)]_1) \in \mathcal{X}_{\mathsf{priv}}^2$	Input: $(\mathbf{v}, [\mathbf{y}_1]_2) \in \mathcal{F}_{\mathsf{pub}}, ([\mathbf{y}_2^0]_2, [\mathbf{y}_2^1]_2) \in \mathcal{F}_{\mathsf{priv}}^2$
$\mathbf{z} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{x}}_1 = (\mathbf{z} \otimes \mathbf{u}, \mathbf{x}_1^\beta),\ \widetilde{\mathbf{x}}_2 = (\mathbf{x}_2^\beta, 0^{m_2}, 0)$	$\mathbf{a} \leftarrow \mathbb{Z}_p^k,\ \widetilde{\mathbf{y}}_1 = (\mathbf{a} \otimes \mathbf{v}, \mathbf{y}_1),\ \widetilde{\mathbf{y}}_2 = (\mathbf{y}_2^\beta, 0^{m_2}, 0)$
$\mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, ([\widetilde{\mathbf{x}}_1]_1, [\widetilde{\mathbf{x}}_2]_1))$	$\mathsf{iSK} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, ([\widetilde{\mathbf{y}}_1]_1, [\widetilde{\mathbf{y}}_2]_1))$
Output: $\mathsf{pCT} = (\mathbf{u}, \mathsf{iCT})$	Output: $\mathsf{pSK} = (\mathbf{v}, \mathsf{iSK})$

Fig 1. The behavior of cO and kO in H_s^β.

- for the j-th query to cO on $(\mathbf{u}^j, ([(\mathbf{x}_1^{j,0}, \mathbf{x}_2^{j,0})]_1, [(\mathbf{x}_1^{j,1}, \mathbf{x}_2^{j,1})]_1))$, it chooses $\mathbf{z}^j \leftarrow \mathbb{Z}_p^k$ and sets $\widetilde{\mathbf{x}}_1^j, \widetilde{\mathbf{x}}_2^j$ as

$$\widetilde{\mathbf{x}}_1^j = \begin{cases} (\mathbf{z}^j \otimes \mathbf{u}^j, \underline{\mathbf{x}_1^{j,0}}) & (j < \iota) \\ (\underline{0^{kd}} \qquad , \mathbf{x}_1^{j,\beta}) & (j = \iota) \\ (\mathbf{z}^j \otimes \mathbf{u}^j, \mathbf{x}_1^{j,\beta}) & (j > \iota) \end{cases}, \quad \widetilde{\mathbf{x}}_2^j = \begin{cases} (\underline{0^{m_2}, \mathbf{x}_2^{j,0}}, 0) & (j < \iota) \\ (\mathbf{x}_2^{j,\beta}, 0^{m_2}, \underline{1}) & (j = \iota) \\ (\mathbf{x}_2^{j,\beta}, 0^{m_2}, 0) & (j > \iota) \end{cases}$$

- for the ℓ-th query to kO on $(\mathbf{v}^\ell, [\mathbf{y}_1^\ell]_2, ([\mathbf{y}_2^{\ell,0}]_2, [\mathbf{y}_2^{\ell,1}]_2))$, it chooses $\mathbf{a}^\ell \leftarrow \mathbb{Z}_p^k$ and sets

$$\widetilde{\mathbf{y}}_1^\ell = (\mathbf{a}^\ell \otimes \mathbf{v}^\ell, \mathbf{y}_1^\ell), \ \ \widetilde{\mathbf{y}}_2^\ell = (\mathbf{y}_2^{\ell,\beta}, \underline{\mathbf{y}_2^{\ell,0}, \langle \mathbf{z}^\iota, \mathbf{a}^\ell \rangle \cdot \langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle})$$

$\mathsf{H}_{\iota,2}^\beta$**:** This game is the same as $\mathsf{H}_{\iota,1}^\beta$ except that in each query to kO, it samples $t^\ell \leftarrow \mathbb{Z}_p$ and sets $\widetilde{\mathbf{y}}_2^\ell = (\mathbf{y}_2^{\ell,\beta}, \mathbf{y}_2^{\ell,0}, \underline{t^\ell} \cdot \langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle)$.

$\mathsf{H}_{\iota,3}^\beta$**:** This game is the same as $\mathsf{H}_{\iota,2}^\beta$ except that cO sets $\widetilde{\mathbf{x}}_1^\iota = (0^{kd}, \underline{\mathbf{x}_1^{\iota,0}})$ and $\widetilde{\mathbf{x}}_2^\iota = (\underline{0^{m_2}, \mathbf{x}_2^{\iota,0}}, 1)$.

$\mathsf{H}_{\iota,4}^\beta$**:** This game is the same as $\mathsf{H}_{\iota,3}^\beta$ except that

- cO sets $\widetilde{\mathbf{x}}_1^\iota = (\underline{\mathbf{z}^\iota \otimes \mathbf{u}^\iota}, \mathbf{x}_1^{\iota,0})$ and $\widetilde{\mathbf{x}}_2^\iota = (0^{m_2}, \mathbf{x}_2^{\iota,0}, \underline{0})$.
- kO sets $\widetilde{\mathbf{y}}_2^\ell = (\mathbf{y}_2^{\ell,\beta}, \mathbf{y}_2^{\ell,0}, \underline{0})$ for all queries.

H_f^β**:** This game is the same as $\mathsf{H}_{q_c,4}^\beta$ except that kO sets $\widetilde{\mathbf{y}}_2^\ell = (\underline{0^{m_2}}, \mathbf{y}_2^{\ell,0}, 0)$ for all queries.

Observe that the adversary does not obtain the information on β in H_f^β, and thus its advantage is 0. Thanks to Lemmata 3.1 to 3.5, Theorem 3.1 holds. $\square$

Lemma 3.1. *Let* $\mathsf{H}_s^\beta = \mathsf{H}_{0,4}^\beta$. *For all* $\iota \in [q_c]$, $\mathsf{H}_{\iota-1,4}^\beta \approx_c \mathsf{H}_{\iota,1}^\beta$ *if* iFE *is IND-partially hiding.*

Proof. Observe that the difference between $\mathsf{H}_{\iota-1,4}^\beta$ and $\mathsf{H}_{\iota,1}^\beta$ is

- $\widetilde{\mathbf{x}}_1^\iota = (\mathbf{z}^\iota \otimes \mathbf{u}^\iota, \mathbf{x}_1^{\iota,\beta}) \longrightarrow \widetilde{\mathbf{x}}_1^\iota = (0^{kd}, \mathbf{x}_1^{\iota,\beta})$
- $\widetilde{\mathbf{x}}_2^\iota = (\mathbf{x}_2^{\iota,\beta}, 0^{m_2}, 0) \longrightarrow \widetilde{\mathbf{x}}_2^\iota = (\mathbf{x}_2^{\iota,\beta}, 0^{m_2}, 1)$
- $\widetilde{\mathbf{y}}_2 = \begin{cases} (\mathbf{y}_2^\beta, 0^{m_2}, 0) & (\iota = 1) \\ (\mathbf{y}_2^\beta, \mathbf{y}_2^0, 0) & (\iota > 1) \end{cases} \longrightarrow \widetilde{\mathbf{y}}_2 = (\mathbf{y}_2^\beta, \mathbf{y}_2^0, \langle \mathbf{z}^\iota, \mathbf{a} \rangle \cdot \langle \mathbf{u}^\iota, \mathbf{v} \rangle)$ for all queries to kO.

For $j \in [q_c]$ and $\ell \in [q_k]$, let $(\widehat{\mathbf{x}}_1^{j,0}, \widehat{\mathbf{x}}_2^{j,0})$ and $(\widehat{\mathbf{y}}_1^{\ell,0}, \widehat{\mathbf{y}}_2^{\ell,0})$ be $(\widetilde{\mathbf{x}}_1^j, \widetilde{\mathbf{x}}_2^j)$ and $(\widetilde{\mathbf{y}}_1^\ell, \widetilde{\mathbf{y}}_2^\ell)$ defined in $\mathsf{H}_{\iota-1,4}^{\beta}$, respectively. Similarly, let $(\widehat{\mathbf{x}}_1^{j,1}, \widehat{\mathbf{x}}_2^{j,1})$ and $(\widehat{\mathbf{y}}_1^{\ell,1}, \widehat{\mathbf{y}}_2^{\ell,1})$ be $(\widetilde{\mathbf{x}}_1^j, \widetilde{\mathbf{x}}_2^j)$ and $(\widetilde{\mathbf{y}}_1^\ell, \widetilde{\mathbf{y}}_2^\ell)$ defined in $\mathsf{H}_{\iota,1}^{\beta}$, respectively. Then, it is not hard to see that $\langle \widehat{\mathbf{x}}_1^{j,0}, \widehat{\mathbf{y}}_1^{\ell,0} \rangle + \langle \widehat{\mathbf{x}}_2^{j,0}, \widehat{\mathbf{y}}_2^{\ell,0} \rangle = \langle \widehat{\mathbf{x}}_1^{j,1}, \widehat{\mathbf{y}}_1^{\ell,1} \rangle + \langle \widehat{\mathbf{x}}_2^{j,1}, \widehat{\mathbf{y}}_2^{\ell,1} \rangle$ and $\widehat{\mathbf{y}}_1^{\ell,0} = \widehat{\mathbf{y}}_1^{\ell,1}$ for all $j \in [q_c]$ and $\ell \in [q_k]$. Thus, we can reduce the indistinguishability between $\mathsf{H}_{\iota-1,4}^{\beta}$ and $\mathsf{H}_{\iota,1}^{\beta}$ to the partially-hiding security of iFE. This concludes the proof. ☐

Lemma 3.2. *For all $\iota \in [q_c]$, $\mathsf{H}_{\iota,1}^{\beta} \approx_c \mathsf{H}_{\iota,2}^{\beta}$ if the MDDH$_k$ assumption holds in $\mathbb{G}$.*

Proof. We can construct an adversary $\mathcal{B}$ against an MDDH$_k$ problem from a distinguisher $\mathcal{A}$ of the two hybrids as follows.

1. $\mathcal{B}$ obtains a $\mathcal{U}_{q_k,k}$-MDDH instance $(\mathbb{G}, [\mathbf{A}]_2, [\mathbf{k}_\delta]_2)$, where $\mathbf{A} \in \mathbb{Z}_p^{q_k \times k}$, $\mathbf{k}_0 = \mathbf{A}\mathbf{z}$, $\mathbf{k}_1 \leftarrow \mathbb{Z}_p^{q_k}$.
2. $\mathcal{B}$ sets $(\mathsf{pPK}, \mathsf{pMSK}) = (\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}$ and gives pPK to $\mathcal{A}$.
3. For all queries to cO, $\mathcal{B}$ replies in the same way as $\mathsf{H}_{\iota,1}^{\beta}$. This is possible since $\mathcal{A}$ generates pMSK by itself.
4. For the ℓ-th query to kO on $(\mathbf{v}^\ell, [\mathbf{y}_1^\ell]_2, ([\mathbf{y}_2^{\ell,0}]_2, [\mathbf{y}_2^{\ell,1}]_2))$, $\mathcal{B}$ replies in the same way as $\mathsf{H}_{\iota,1}^{\beta}$ except that it sets $\widetilde{\mathbf{y}}_2^\ell = (\mathbf{y}_2^{\ell,\beta}, \mathbf{y}_2^{\ell,0}, k_\ell \cdot \langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle)$, where $\mathbf{a}^\ell$ is the ℓ-th row of $\mathbf{A}$ and k_ℓ is the ℓ-th entry of $\mathbf{k}_\delta$.
5. $\mathcal{B}$ outputs 1 if $\mathcal{A}$ outputs β, and outputs 0 otherwise.

It is not hard to see that $\mathcal{A}$'s view corresponds to $\mathsf{H}_{\iota,1}^{\beta}$ if $\delta = 0$ and $\mathsf{H}_{\iota,2}^{\beta}$ otherwise. ☐

Lemma 3.3. *For all $\iota \in [q_c]$, $\mathsf{H}_{\iota,2}^{\beta} \approx_c \mathsf{H}_{\iota,3}^{\beta}$ if iFE is IND-partially hiding.*

Proof. For $j \in [q_c]$ and $\ell \in [q_k]$, let $(\widehat{\mathbf{x}}_1^{j,0}, \widehat{\mathbf{x}}_2^{j,0})$ and $(\widehat{\mathbf{y}}_1^{\ell,0}, \widehat{\mathbf{y}}_2^{\ell,0})$ be $(\widetilde{\mathbf{x}}_1^j, \widetilde{\mathbf{x}}_2^j)$ and $(\widetilde{\mathbf{y}}_1^\ell, \widetilde{\mathbf{y}}_2^\ell)$ defined in $\mathsf{H}_{\iota,2}^{\beta}$, respectively. Similarly, let $(\widehat{\mathbf{x}}_1^{j,1}, \widehat{\mathbf{x}}_2^{j,1})$ and $\widehat{\mathbf{y}}_1^{\ell,1}$ be $(\widetilde{\mathbf{x}}_1^j, \widetilde{\mathbf{x}}_2^j)$ and $\widetilde{\mathbf{y}}_1^\ell$ defined in $\mathsf{H}_{\iota,3}^{\beta}$, respectively. Let

$$\widehat{\mathbf{y}}_2^{\ell,1} = (\mathbf{y}_2^{\ell,\beta}, \mathbf{y}_2^{\ell,0}, t^\ell \cdot \langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle + \langle \mathbf{x}_1^{\iota,\beta} - \mathbf{x}_1^{\iota,0}, \mathbf{y}_1^\ell \rangle + \langle \mathbf{x}_2^{\iota,\beta}, \mathbf{y}_2^{\ell,\beta} \rangle - \langle \mathbf{x}_2^{\iota,0}, \mathbf{y}_2^{\ell,0} \rangle).$$

Then, it is not hard to see that $\langle \widehat{\mathbf{x}}_1^{j,0}, \widehat{\mathbf{y}}_1^{\ell,0} \rangle + \langle \widehat{\mathbf{x}}_2^{j,0}, \widehat{\mathbf{y}}_2^{\ell,0} \rangle = \langle \widehat{\mathbf{x}}_1^{j,1}, \widehat{\mathbf{y}}_1^{\ell,1} \rangle + \langle \widehat{\mathbf{x}}_2^{j,1}, \widehat{\mathbf{y}}_2^{\ell,1} \rangle$ and $\widehat{\mathbf{y}}_1^{\ell,0} = \widehat{\mathbf{y}}_1^{\ell,1}$ for all $j \in [q_c]$ and $\ell \in [q_k]$. Thus, we can reduce the indistinguishability between the 0-side and 1-side to the function-hiding property of iFE. Here, we have the two cases:

$\langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle = 0$: Due to the admissibility of $\mathcal{A}$, we have

$$\langle \mathbf{x}_1^{\iota,\beta} - \mathbf{x}_1^{\iota,0}, \mathbf{y}_1^\ell \rangle + \langle \mathbf{x}_2^{\iota,\beta}, \mathbf{y}_2^{\ell,\beta} \rangle - \langle \mathbf{x}_2^{\iota,0}, \mathbf{y}_2^{\ell,0} \rangle = 0$$

$\langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle \neq 0$: Since t_ℓ is distributed randomly in $\mathbb{Z}_p$, the term $t_\ell \cdot \langle \mathbf{u}^\iota, \mathbf{v}^\ell \rangle$ is also distributed randomly.

Hence, $\widehat{\mathbf{y}}_2^{\ell,0}$ and $\widehat{\mathbf{y}}_2^{\ell,1}$ are identically distributed in both cases, which means that the 0-side corresponds to $\mathsf{H}_{\iota,2}^{\beta}$ and the 1-side corresponds to $\mathsf{H}_{\iota,3}^{\beta}$. ☐

Lemma 3.4. *For all $\iota \in [q_c]$, $\mathsf{H}^{\beta}_{\iota,3} \approx_c \mathsf{H}^{\beta}_{\iota,4}$ if iFE is IND-partially hiding and the MDDH$_k$ assumption holds in $\mathbb{G}$.*

We omit the proof since Lemma 3.4 can be proven similarly to Lemmata 3.1 and 3.2.

Lemma 3.5. *$\mathsf{H}^{\beta}_{q_c,4} \approx_c \mathsf{H}^{\beta}_{f}$ if iFE is IND-partially hiding.*

We omit the proof since Lemma 3.5 can be proven similarly to Lemma 3.1.

4 Unbounded Slotted Inner Product Functional Encryption

In this section, we define a new primitive called unbounded slotted IPFE and show how to construct it. We use it as a building block of our FE schemes for unbounded quadratic functions (Section 5) and ABP $\circ$ UQF (Section 6).

4.1 Definitions

Definition 4.1 (Unbounded Slotted IPFE). Let $\mathbb{G}$ be bilinear groups, $\mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv1}} \times \mathcal{X}_{\mathsf{priv2}} = \{(x_1, x_2, x_3) \in 2^{[p]} \times \bigcup_{i \in [p]} (G_1^{m_1})^i \times G_1^{m_2} \mid |x_1| = |x_2|/m_1\}$, where $|x_1|$ denotes the cardinality of x_1, and $|x_2|$ denotes the length of x_2. Let $\mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}} = \{((f_1, f_2), f_3) \in (2^{[p]} \times \bigcup_{i \in [p]} (G_2^{m_1})^i) \times G_2^{m_2} \mid |f_1| = |f_2|/m_1\}$. A function family $\mathcal{F}^{\mathsf{UIP}}_{m_1,m_2,\mathbb{G}} = \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ consists of functions $f : \mathcal{X}_{\mathsf{pub}} \times \mathcal{X}_{\mathsf{priv}} \to G_T \cup \{\perp\}$. Each $f \in \mathcal{F}^{\mathsf{UIP}}_{m_1,m_2,\mathbb{G}}$ is specified by $((S_k, [\mathbf{y}]_2), [\mathbf{y}_0]_2) \in \mathcal{F}_{\mathsf{pub}} \times \mathcal{F}_{\mathsf{priv}}$ where $S_k \subseteq [p], \mathbf{y} = (\mathbf{y}_i)_{i \in S_k} \in (\mathbb{Z}_p^{m_1})^{S_k}, \mathbf{y}_0 \in \mathbb{Z}_p^{m_2}$ and defined as

$$f(S_c, [\mathbf{x}]_1, [\mathbf{x}_0]_1) = \begin{cases} [\sum_{i \in S_k} \langle \mathbf{x}_i, \mathbf{y}_i \rangle + \langle \mathbf{x}_0, \mathbf{y}_0 \rangle]_T & \text{if } S_k \subseteq S_c \\ \perp & \text{otherwise} \end{cases}$$

where $S_c \subseteq [p], \mathbf{x} = (\mathbf{x}_i)_{i \in S_c} \in (\mathbb{Z}_p^{m_1})^{S_c}, \mathbf{x}_0 \in \mathbb{Z}_p^{m_2}$. Note that S_c is the public input while $[\mathbf{x}]_1$ is a private input for the first slot, and $[\mathbf{x}_0]_1$ is a private input for the second slot. We refer to slotted FE (Definition 2.5) for $\mathcal{F}^{\mathsf{UIP}}_{m_1,m_2,\mathbb{G}}$ as unbounded slotted IPFE.

4.2 Unbounded Slotted IPFE from Predicate Slotted IPFE

Construction. Let k be the parameter of the MDDH$_k$ assumption. Let $\mathsf{pFE} = (\mathsf{pSetup}, \mathsf{pEnc}, \mathsf{pSlotEnc}, \mathsf{pKeyGen}, \mathsf{pDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{PIP}}_{2,m_1+k,1,\mathbb{G}}$ with slot-mode correctness for $e = [0]_1$. Let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{IP}}_{k,m_2+1,\mathbb{G}}$ with slot-mode correctness for $e = [0^{m_2+1}]_1$. Then, our partially hiding slotted FE scheme $\mathsf{uFE} = (\mathsf{uSetup}, \mathsf{uEnc}, \mathsf{uSlotEnc}, \mathsf{uKeyGen}, \mathsf{uDec})$ for $\mathcal{F}^{\mathsf{UIP}}_{m_1,m_2,\mathbb{G}}$ with slot-mode correctness for $e = [0^{m_2}]_1$ is constructed as follows.

$\mathsf{uSetup}(1^\lambda)$**:** It runs $(\mathsf{pPK}, \mathsf{pMSK}) \leftarrow \mathsf{pSetup}(1^\lambda)$, $(\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$, and outputs $(\mathsf{uPK}, \mathsf{uMSK}) = ((\mathsf{pPK}, \mathsf{iPK}), (\mathsf{pMSK}, \mathsf{iMSK}))$.

$\mathsf{uEnc}(\mathsf{uMSK}, (S_c, [\mathbf{x}]_1, [\mathbf{x}_0]_1))$**:** It chooses $\mathbf{z} \leftarrow \mathbb{Z}_p^k$ and outputs uCT as follows:

$$\mathbf{u}_i = (1, i),\ \widetilde{\mathbf{x}}_i = (\mathbf{x}_i, \mathbf{z}, 0),\ \mathsf{pCT}_i \leftarrow \mathsf{pEnc}(\mathsf{pMSK}, (\mathbf{u}_i, [\widetilde{\mathbf{x}}_i]_1)) \text{ for } i \in S_c$$
$$\widetilde{\mathbf{x}}_0 = (\mathbf{z}, \mathbf{x}_0, 0),\ \mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\widetilde{\mathbf{x}}_0]_1),\ \mathsf{uCT} = (S_c, \{\mathsf{pCT}_i\}_{i \in S_c}, \mathsf{iCT}).$$

$\mathsf{uSlotEnc}(S_c, [\mathbf{x}]_1)$**:** It chooses $\mathbf{z} \leftarrow \mathbb{Z}_p^k$ and outputs uCT as follows:

$$\mathbf{u}_i = (1, i),\ \widetilde{\mathbf{x}}_i = (\mathbf{x}_i, \mathbf{z}),\ \mathsf{pCT}_i \leftarrow \mathsf{pSlotEnc}(\mathbf{u}_i, [\widetilde{\mathbf{x}}_i]_1) \text{ for } i \in S_c$$
$$\mathsf{iCT} \leftarrow \mathsf{iSlotEnc}([\mathbf{z}]_1),\ \mathsf{uCT} = (S_c, \{\mathsf{pCT}_i\}_{i \in S_c}, \mathsf{iCT}).$$

$\mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\mathbf{y}]_2, [\mathbf{y}_0]_2))$**:** It chooses $\mathbf{a}_i \leftarrow \mathbb{Z}_p^k$ for all $i \in S_k$, sets $\mathbf{a}_0 = -\sum_{i \in S_k} \mathbf{a}_i$, and outputs uSK as follows:

$$\mathbf{v}_i = (i, -1),\ \widetilde{\mathbf{y}}_i = (\mathbf{y}_i, \mathbf{a}_i, 0),\ \mathsf{pSK}_i \leftarrow \mathsf{pKeyGen}(\mathsf{pMSK}, (\mathbf{v}_i, [\widetilde{\mathbf{y}}_i]_2)) \text{ for } i \in S_k$$
$$\widetilde{\mathbf{y}}_0 = (\mathbf{a}_0, \mathbf{y}_0, 0),\ \mathsf{iSK} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{y}}_0]_2),\ \mathsf{uSK} = (S_k, \{\mathsf{pSK}_i\}_{i \in S_k}, \mathsf{iSK}).$$

$\mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK})$**:** If $S_k \not\subseteq S_c$, it outputs $\perp$. Otherwise, outputs $\mathsf{iDec}(\mathsf{iCT}, \mathsf{iSK}) + \sum_{i \in S_k} \mathsf{pDec}(\mathsf{pCT}_i, \mathsf{pSK}_i)$.

Correctness. Thanks to the correctness of iFE and pFE, $\mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK})$ outputs $[\sum_{i \in S_k} (\langle \mathbf{x}_i, \mathbf{y}_i \rangle + \langle \mathbf{z}, \mathbf{a}_i \rangle) + \langle \mathbf{x}_0, \mathbf{y}_0 \rangle + \langle \mathbf{z}, \mathbf{a}_0 \rangle]_T = [\sum_{i \in S_k} \langle \mathbf{x}_i, \mathbf{y}_i \rangle + \langle \mathbf{x}_0, \mathbf{y}_0 \rangle]_T$.
Slot-mode correctness. Thanks to slot-mode correctness of pFE, $\mathsf{pSlotEnc}(\mathbf{u}_i, [\widetilde{\mathbf{x}}_1]_1)$ and $\mathsf{pEnc}(\mathsf{pMSK}, (\mathbf{u}_i, [(\widetilde{\mathbf{x}}_1, 0)]_1))$ are identically distributed for all correctly generated $(\mathsf{pMSK}, \mathsf{pPK})$, $\mathbf{u}_i \in \mathbb{Z}_p^2$, and $\widetilde{\mathbf{x}}_1 \in \mathbb{Z}_p^{m_1+k}$. Similarly, $\mathsf{iSlotEnc}([\mathbf{z}]_1)$ and $\mathsf{iEnc}(\mathsf{iMSK}, ([(\mathbf{z}, 0^{m_2+1})]_1))$ are identically distributed for all correctly generated $(\mathsf{iMSK}, \mathsf{iPK})$ and $\mathbf{x} \in \mathbb{Z}_p^k$. Hence, $\mathsf{uSlotEnc}(S_c, [\mathbf{x}]_1)$ and $\mathsf{uEnc}(\mathsf{uMSK}, (S_c, [\mathbf{x}]_1, [0^{m_2}]_1))$ are identically distributed for all correctly generated $(\mathsf{uMSK}, \mathsf{uPK})$, $S_c \subseteq [p]$, and $\mathbf{x} \in (\mathbb{Z}_p^{m_1})^{S_c}$.

4.3 Security Analysis

For security, we have the following theorem.

Theorem 4.1. *If* pFE *and* iFE *are IND-partially hiding, and the MDDH$_k$ assumption holds in* $\mathbb{G}$*, then* uFE *is IND-1-partially hiding.*

Proof. We prove Theorem 4.1 via a series of hybrid games $\mathsf{H}_1^\beta, \mathsf{H}_2^\beta, \mathsf{H}_f^\beta$. We show that $\mathsf{H}_s^\beta \approx_c \mathsf{H}_1^\beta \approx_c \mathsf{H}_2^\beta \approx_c \mathsf{H}_f^\beta$, where H_s^β for $\beta \in \{0, 1\}$ is the original security game (described in Eq. (2.3)). Especially, the oracles cO and kO works as Fig 2 in H_s^β. In the hybrid sequence, the behavior of the oracles is gradually changed. Each hybrid is defined as follows.

H_1^β**:** This game is the same as H_s^β except that

$\underline{\mathsf{cO}(\beta,\cdot)}$	$\underline{\mathsf{kO}(\beta,\cdot)}$
Input: $S_c \in \mathcal{X}_{\mathsf{pub}}, ([\mathbf{x}^0]_1, [\mathbf{x}^1]_1) \in \mathcal{X}^2_{\mathsf{priv1}}, ([\mathbf{x}^0_0]_1, [\mathbf{x}^1_0]_1) \in \mathcal{X}^2_{\mathsf{priv2}}$	Input: $(S_k, [\mathbf{y}]_2) \in \mathcal{F}_{\mathsf{pub}}, ([\mathbf{y}^0_0]_2, [\mathbf{y}^1_0]_2) \in \mathcal{F}^2_{\mathsf{priv}}$
$\mathbf{z} \leftarrow \mathbb{Z}_p^k,\ \mathbf{u}_i = (1, i)$	$\mathbf{a}_i \leftarrow \mathbb{Z}_p^k,\ \mathbf{a}_0 = -\sum_{i\in S_k} \mathbf{a}_i,\ \mathbf{v}_i = (i, -1)$
$\widetilde{\mathbf{x}}_i = (\mathbf{x}_i^\beta, \mathbf{z}, 0),\ \mathsf{pCT}_i \leftarrow \mathsf{pEnc}(\mathsf{pMSK}, (\mathbf{u}_i, [\widetilde{\mathbf{x}}_i]_1))$	$\widetilde{\mathbf{y}}_i = (\mathbf{y}_i, \mathbf{a}_i, 0),\ \mathsf{pSK}_i \leftarrow \mathsf{pKeyGen}(\mathsf{pMSK}, (\mathbf{v}_i, [\widetilde{\mathbf{y}}_i]_2))$
$\widetilde{\mathbf{x}}_0 = (\mathbf{z}, \mathbf{x}_0^\beta, 0),\ \mathsf{iCT} \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\widetilde{\mathbf{x}}_0]_1)$	$\widetilde{\mathbf{y}}_0 = (\mathbf{a}_0, \mathbf{y}_0^\beta, 0),\ \mathsf{iSK} \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{y}}_0]_2)$
Output: $\mathsf{uCT} = (S_c, \{\mathsf{pCT}_i\}_{i\in S_c}, \mathsf{iCT})$	Output: $\mathsf{uSK} = (S_k, \{\mathsf{pSK}_i\}_{i\in S_k}, \mathsf{iSK})$.

Fig 2. The behavior of cO and kO in H_s^β.

– for the query to cO, it chooses $\mathbf{z} \leftarrow \mathbb{Z}_p^k$ and sets $\widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_0^j$ as

$$\widetilde{\mathbf{x}}_i = (\underline{0^{m_1}, 0^k, 1}),\quad \widetilde{\mathbf{x}}_0 = (\underline{0^k, 0^{m_2}, 1})$$

– for the ℓ-th query to kO on $(S_k^\ell, [\mathbf{y}^\ell]_2, ([\mathbf{y}_0^{\ell,0}]_2, [\mathbf{y}_0^{\ell,1}]_2))$, it chooses $\mathbf{a}_i^\ell \leftarrow \mathbb{Z}_p^k$ for $i \in S_k^\ell$ and sets $\mathbf{a}_0^\ell = -\sum_{i\in S_k} \mathbf{a}_i^\ell$ and

$$\widetilde{\mathbf{y}}_i^\ell = \begin{cases} (\mathbf{y}_i^\ell, \mathbf{a}_i^\ell, \underline{\langle \mathbf{x}_i^\beta, \mathbf{y}_i^\ell\rangle + \langle \mathbf{z}, \mathbf{a}_i^\ell\rangle}) & (i \in S_c) \\ (\mathbf{y}_i^\ell, \mathbf{a}_i^\ell, 0) & (i \notin S_c) \end{cases}$$

$$\widetilde{\mathbf{y}}_0^\ell = (\mathbf{a}_0^\ell, \underline{\mathbf{y}_0^{\ell,0}, \langle \mathbf{z}, \mathbf{a}_0^\ell\rangle + \langle \mathbf{x}_0^\beta, \mathbf{y}_0^{\ell,\beta}\rangle})$$

H_2^β**:** This game is the same as H_1^β except the following: in each query to kO, it samples $t_i^\ell \leftarrow \mathbb{Z}_p$ for $i \in S_k^\ell \cup \{0\}$ so that $\sum_{i\in S_k^\ell\cup\{0\}} t_i^\ell = 0$ if $S_k^\ell \subseteq S_c$, and otherwise it just randomly samples $t_i^\ell \leftarrow \mathbb{Z}_p$ for $i \in (S_c \cap S_k^\ell) \cup \{0\}$. Then, it sets

$$\widetilde{\mathbf{y}}_i^\ell = \begin{cases} (\mathbf{y}_i^\ell, \mathbf{a}_i^\ell, \langle \mathbf{x}_i^\beta, \mathbf{y}_i^\ell\rangle + \underline{t_i^\ell}) & (i \in S_c) \\ (\mathbf{y}_i^\ell, \mathbf{a}_i^\ell, 0) & (i \notin S_c) \end{cases},\quad \widetilde{\mathbf{y}}_0^\ell = (\mathbf{a}_0^\ell, \mathbf{y}_0^{\ell,0}, \underline{t_0^\ell} + \langle \mathbf{x}_0^\beta, \mathbf{y}_0^{\ell,\beta}\rangle)$$

H_f^β**:** This game is the same as H_2^β except the following: it sets

$$\widetilde{\mathbf{y}}_i^\ell = \begin{cases} (\mathbf{y}_i^\ell, \mathbf{a}_i^\ell, \langle \underline{\mathbf{x}_i^0}, \mathbf{y}_i^\ell\rangle + t_i^\ell) & (i \in S_c) \\ (\mathbf{y}_i^\ell, \mathbf{a}_i^\ell, 0) & (i \notin S_c) \end{cases},\quad \widetilde{\mathbf{y}}_0^\ell = (\mathbf{a}_0^\ell, \mathbf{y}_0^{\ell,0}, t_0^\ell + \underline{\langle \mathbf{x}_0^0, \mathbf{y}_0^{\ell,0}\rangle})$$

Observe that the adversary does not obtain the information on β in H_f^β, and thus its advantage is 0. Thanks to Lemmata **??** to **??**, Theorem 4.1 holds. □

5 Unbounded Quadratic Functional Encryption

In this section, we present our FE scheme for unbounded quadratic functions defined in Definition 2.7.

5.1 Construction

Let k be the parameter of the MDDH$_k$ assumption. Let $\mathsf{uFE} = (\mathsf{uSetup}, \mathsf{uEnc}, \mathsf{uSlotEnc}, \mathsf{uKeyGen}, \mathsf{uDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{UIP}}_{k,1,\mathbb{G}}$ with slot-mode correctness for $e = [0]_1$. Let $H : [p] \to G_2^k$ be a hash function modeled as a random oracle. Then, our partially hiding FE scheme $\mathsf{qFE} = (\mathsf{qSetup}, \mathsf{qEnc}, \mathsf{qKeyGen}, \mathsf{qDec})$ for $\mathcal{F}^{\mathsf{UQF}}_{\mathbb{G}}$ is constructed as follows.

$\mathsf{qSetup}(1^\lambda)$**:** It runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uKeyGen}(1^\lambda)$ outputs $(\mathsf{qPK}, \mathsf{qMSK}) = (\mathsf{uPK}, \mathsf{uMSK})$.

$\mathsf{qEnc}(S_c, \mathbf{x} = (x_i)_{i \in S_c})$**:** First, it defines vectors as follows:

$$[\mathbf{a}_i]_2 = H(i),\ \mathbf{z}_i \leftarrow \mathbb{Z}_p^k,\ \mathbf{b}_i = (x_i, \mathbf{z}_i, 0),\ \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, 0)$$
$$\mathbf{d}_i = \mathbf{z}_i,\ \mathbf{d} = (\mathbf{d}_i)_{i \in S_c}.$$

Then, it outputs qCT as follows: let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{IP}}_{0,k+2,\mathbb{G}}$ with slot-mode correctness for $e = [0^{k+2}]_1$, or equivalently standard function-hiding IPFE scheme with the vector length being $k+2$.

$$(\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$$
$$\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{b}_i]_1),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{b}}_i]_2)$$
$$\mathsf{uCT} \leftarrow \mathsf{uSlotEnc}(S_c, [\mathbf{d}]_1),\ \mathsf{qCT} = (\mathsf{iPK}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in S_c}, \mathsf{uCT})$$

$\mathsf{qKeyGen}(\mathsf{qMSK}, (S_k, \mathbf{c} = (c_{i,j})_{i,j \in S_k}))$**:** It outputs qSK as follows:

$$[\mathbf{a}_j]_2 = H(j),\ \widetilde{\mathbf{d}}_i = \sum_{j \in S_k} c_{i,j}\mathbf{a}_j,\ \widetilde{\mathbf{d}} = (\widetilde{\mathbf{d}}_i)_{i \in S_k}$$
$$\mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [0]_2)),\ \mathsf{qSK} = \mathsf{uSK}$$

$\mathsf{qDec}(\mathsf{qCT}, \mathsf{qSK})$**:** If $S_k \not\subseteq S_c$, it outputs $\perp$. Otherwise, it outputs $[z]_T$ as follows:

$$[z_1]_T = \sum_{i,j \in S_k} c_{i,j}\mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j),\ [z_2]_T = \mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK})$$
$$[z]_T = [z_1 - z_2]_T.$$

Correctness. Due to the correctness of iFE and uEF, we have

$$z_1 = \sum_{i,j \in S_k} (c_{i,j}x_i x_j + c_{i,j}\langle \mathbf{z}_i, \mathbf{a}_j \rangle),\ z_2 = \sum_{i,j \in S_k} c_{i,j}\langle \mathbf{z}_i, \mathbf{a}_j \rangle$$

Hence, we have $z = \sum_{i,j \in S_k} c_{i,j}x_i x_j$.

5.2 Security

For security, we have the following theorem.

Theorem 5.1. *If* iFE *is IND-partially hiding,* uFE *is IND-1-partially hiding, and the MDDH$_k$ assumption holds in $\mathbb{G}$, then* qFE *is SIM-partially-hiding.*

Proof. First, we show our simulation algorithms. Note that our simulation algorithm for key key generation takes a G_2 element instead of a G_T element, which follows [Wee20, GQ20].

$\underline{\mathsf{H}_s}$	$\underline{\mathsf{H}_\eta}$
$\mathsf{qPK}, \mathsf{qMSK} \leftarrow \mathsf{qSetup}(1^\lambda)$	$\mathsf{qPK}, \mathsf{qMSK} \leftarrow \mathsf{qSetup}(1^\lambda)$
$\widetilde{\mathbf{x}} = (S_c, \mathbf{x}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{qPK})$	$\widetilde{\mathbf{x}} = (S_c, \mathbf{x}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{qPK})$
$\mathsf{qCT} \leftarrow \mathsf{qEnc}(\widetilde{\mathbf{x}})$	$\mathsf{qCT} \leftarrow \widetilde{\mathsf{qEnc}}_\eta(\mathsf{qMSK}, \widetilde{\mathbf{x}})$
$\beta \leftarrow \mathcal{A}^{\mathsf{qKeyGen}(\mathsf{qMSK}, \cdot)}(\mathsf{qCT})$	$\beta \leftarrow \mathcal{A}^{\widetilde{\mathsf{qKeyGen}}_\eta(\mathsf{qMSK}, \widetilde{\mathbf{x}}, \cdot)}(\mathsf{qCT})$

Fig 3. Hybrids for qFE.

$\mathsf{qSetup}^*(1^\lambda)$**:** It runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uKeyGen}(1^\lambda)$ outputs $(\mathsf{qPK}^*, \mathsf{qMSK}^*) = (\mathsf{uPK}, \mathsf{uMSK})$.

$\mathsf{qEnc}^*(\mathsf{qMSK}^*, S_c)$**:** First, it defines vectors as follows:

$$[\mathbf{a}_i]_2 = H(i),\ \mathbf{z}_i \leftarrow \mathbb{Z}_p^k,\ \mathbf{b}_i = (0, \mathbf{z}_i, 0),\ \widetilde{\mathbf{b}}_i = (0, \mathbf{a}_i, 0)$$
$$\mathbf{d}_i = \mathbf{z}_i,\ \mathbf{d} = (\mathbf{d}_i)_{i \in S_c}.$$

Then, it outputs qCT^* as follows: let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{IP}}_{0,k+2,\mathbb{G}}$ with slot-mode correctness for $e = [0^{k+2}]_1$.

$$(\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$$
$$\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{b}_i]_1),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{b}}_i]_2)$$
$$\mathsf{uCT} \leftarrow \mathsf{uEnc}(\mathsf{uMSK}, (S_c, [\mathbf{d}]_1, [1]_1)),\ \mathsf{qCT}^* = (\mathsf{iPK}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in S_c}, \mathsf{uCT})$$

$\mathsf{qKeyGen}^*(\mathsf{qMSK}^*, (S_k, \mathbf{c} = (c_{i,j})_{i,j \in S_k}, S_c, [\alpha]_2 \text{ or } \perp))$**:** It outputs qSK^* as follows:

$$[\mathbf{a}_j]_2 = H(j),\ \widetilde{\mathbf{d}}_i = \sum_{j \in S_k} c_{i,j}\mathbf{a}_j,\ \widetilde{\mathbf{d}} = (\widetilde{\mathbf{d}}_i)_{i \in S_k}$$
$$\mathsf{uSK} \leftarrow \begin{cases} \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [-\alpha]_2)) & S_k \subseteq S_c \\ \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [0]_2)) & \text{otherwise} \end{cases},\quad \mathsf{qSK}^* = \mathsf{uSK}$$

We prove Theorem 5.1 via a series of hybrid games H_η for $\eta \in [s_{\mathsf{max}}] \cup \{f\}$ where s_{max} is the maximum size of the challenge index set S_c. We show that $\mathsf{H}_s \approx_c \mathsf{H}_1 \approx_c \cdots \approx_c \mathsf{H}_{s_{\mathsf{max}}} \approx_c \mathsf{H}_f$, where H_s is the real game. Each hybrid is defined as described in Fig 3, where qEnc and $\mathsf{qKeyGen}$ are replaced with $\widetilde{\mathsf{qEnc}}_\eta$ and $\widetilde{\mathsf{qKeyGen}}_\eta$. They work as follows for $\eta \in [s_{\mathsf{max}}]$.

$\widetilde{\mathsf{qEnc}}_\eta(\mathsf{qMSK}, \widetilde{\mathbf{x}})$**:** Let $S_c = (s_1, \ldots, s_{|S_c|})$. First, it defines vectors as follows:

$$[\mathbf{a}_i]_2 = H(i),\ \mathbf{z}_i \leftarrow \mathbb{Z}_p^k$$
$$\mathbf{b}_i = \begin{cases} (\underline{0},\ \mathbf{z}_i, 0) & (i \le s_\eta) \\ (x_i, \mathbf{z}_i, 0) & (i > s_\eta) \end{cases},\quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, 0) \tag{5.1}$$
$$\mathbf{d}_i = \mathbf{z}_i,\ \mathbf{d} = (\mathbf{d}_i)_{i \in S_c} \tag{5.2}$$

Then, it outputs qCT as follows:

$$
\begin{aligned}
&(\mathsf{iPP}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda) \\
&\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{b}_i]_1), \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{b}}_i]_2) \\
&\mathsf{uCT} \leftarrow \underline{\mathsf{uEnc}(\mathsf{uMSK}, (S_c, [\mathbf{d}]_1, [1]_1))},\ \mathsf{qCT} = (\mathsf{iPP}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in S_c}, \mathsf{uCT})
\end{aligned}
$$

$\widetilde{\mathsf{qKeyGen}}_\eta(\mathsf{qMSK}, \widetilde{\mathbf{x}}, (S_k, \mathbf{c}))$**:** Let $S_{c,\eta} = (s_1, \ldots, s_\eta)$ where s_i is the i-th element of the challenge index set S_c. It outputs qSK as follows:

$$
\begin{aligned}
&[\mathbf{a}_j]_2 = H(j),\ \widetilde{\mathbf{d}}_i = \sum_{j \in S_k} c_{i,j} \mathbf{a}_j,\ \widetilde{\mathbf{d}} = (\widetilde{\mathbf{d}}_i)_{i \in S_k} \\
&\widehat{d} = \begin{cases} \underline{- \sum_{\substack{i \in S_{c,\eta} \cap S_k \\ j \in S_k}} c_{i,j} x_i x_j} & S_k \subseteq S_c \\ 0 & \text{otherwise} \end{cases} \\
&\mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, \underline{[\widehat{d}]_2})),\ \mathsf{qSK} = \mathsf{uSK}
\end{aligned}
$$

H_f is the same as $\mathsf{H}_{s_{\mathsf{max}}}$ except that $\widetilde{\mathsf{qEnc}}_\eta$ sets $\widetilde{\mathbf{b}}_i = (\underline{0}, \mathbf{a}_i, 0)$ in Eq. (5.1). Observe that the adversary's view in H_f is equivalent to that in the simulated game. Thanks to Lemmata 5.1 and 5.2, Theorem 5.1 holds. □

Lemma 5.1. $\mathsf{H}_{s_{\mathsf{max}}} \approx_c \mathsf{H}_f$ *if* iFE *is IND-partially hiding.*

Proof. For all $i \in [|S_c|]$, let $\mathbf{b}_i^0$ and $\widetilde{\mathbf{b}}_i^0$ be $\mathbf{b}_i$ and $\widetilde{\mathbf{b}}_i$ defined in $\mathsf{H}_{s_{\mathsf{max}}}$. Similarly, let $\mathbf{b}_i^1$ and $\widetilde{\mathbf{b}}_i^1$ be $\mathbf{b}_i$ and $\widetilde{\mathbf{b}}_i$ defined in H_f. Then, it is not hard to see that $\langle \mathbf{b}_i^0, \widetilde{\mathbf{b}}_j^0 \rangle = \langle \mathbf{b}_i^1, \widetilde{\mathbf{b}}_j^1 \rangle$ for all $i, j \in [|S_c|]$. Hence, the difference between $\mathsf{H}_{s_{\mathsf{max}}}$ and H_f can be reduced to partially hiding security of iFE. □

Lemma 5.2. *Let* $\mathsf{H}_s = \mathsf{H}_0$*. For* $\eta \in [s_{\mathsf{max}}]$*, we have* $\mathsf{H}_{\eta-1} \approx_c \mathsf{H}_\eta$ *if* iFE *and* uFE *are IND-partially hiding and the MDDH*$_k$ *assumption holds in* $\mathbb{G}$*.*

Proof. We define intermediate hybrids $\widehat{\mathsf{H}}_{\eta,1}, \widehat{\mathsf{H}}_{\eta,2}, \widehat{\mathsf{H}}_{\eta,3}$ and prove that $\mathsf{H}_{\eta-1} \approx_c \widehat{\mathsf{H}}_{\eta,1} \approx_c \widehat{\mathsf{H}}_{\eta,2} \approx_c \widehat{\mathsf{H}}_{\eta,3} \approx_c \mathsf{H}_\eta$. $\widehat{\mathsf{H}}_{\eta,i}$ for $i \in \{1,2,3\}$ is the same as $\mathsf{H}_{\eta-1}$ except that $\widetilde{\mathsf{qEnc}}_{\eta-1}, \widetilde{\mathsf{qKeyGen}}_{\eta-1}$ are replaced by $\widehat{\mathsf{qEnc}}_{\eta,i}, \widehat{\mathsf{qKeyGen}}_{\eta,i}$, respectively, which work as follows:

$\widehat{\mathsf{qEnc}}_{\eta,1}(\mathsf{qMSK}, \widetilde{\mathbf{x}})$**:** It is the same as $\widetilde{\mathsf{qEnc}}_{\eta-1}$ except that it defines vectors as follows:

$$
\mathbf{b}_i = \begin{cases} (0,\ \mathbf{z}_i, 0) & (i < s_\eta) \\ \underline{(0,\ \mathbf{0},\ 1)} & (i = s_\eta) \\ (x_i, \mathbf{z}_i, 0) & (i > s_\eta) \end{cases},\quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, \underline{\langle \mathbf{z}_{s_\eta}, \mathbf{a}_i \rangle + x_{s_\eta} x_i}) \tag{5.3}
$$

$$
\mathbf{d}_i = \begin{cases} \underline{\mathbf{0}} & i = s_\eta \\ \mathbf{z}_i & i \neq s_\eta \end{cases},\quad \mathbf{d} = (\mathbf{d}_i)_{i \in S_c} \tag{5.4}
$$

$\widehat{\mathsf{qKeyGen}}_{\eta,1}(\mathsf{qMSK}, \widetilde{\mathbf{x}}, (S_k, \mathbf{c}))$**:** It is the same as $\widetilde{\mathsf{qEnc}}_{\eta-1}$ except that, if and only if $S_k \subseteq S_c$, it defines uSK as follows:

$$\mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [\widehat{d} + \underline{\sum_{i \in S_k} c_{s_\eta,i} \langle \mathbf{z}_{s_\eta}, \mathbf{a}_i \rangle}]_2))$$

where $c_{s_\eta,i} = 0$ if $s_\eta \notin S_k$.

$\widehat{\mathsf{qEnc}}_{\eta,2}(\mathsf{qMSK}, \widetilde{\mathbf{x}})$**:** It is the same as $\widehat{\mathsf{qEnc}}_{\eta,1}$ except that it defines vectors as follows:

$$\underline{\mathbf{r} = (r_i)_{i \in S_c} \leftarrow \mathbb{Z}_p^{S_c}}, \quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, \underline{r_i} + x_{s_\eta} x_i)$$

$\widehat{\mathsf{qKeyGen}}_{\eta,2}(\mathsf{qMSK}, \widetilde{\mathbf{x}}, (S_k, \mathbf{c}))$**:** Let $\mathbf{r} = (r_i)_{i \in S_c}$ be the random vector chosen in $\widehat{\mathsf{qEnc}}_{\eta,2}$. It is the same as $\widehat{\mathsf{qKeyGen}}_{\eta,1}$ except that, if and only if $S_k \subseteq S_c$, it defines uSK as follows:

$$\mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [\widehat{d} + \underline{\sum_{i \in S_k} c_{s_\eta,i} r_i}]_2))$$

$\widehat{\mathsf{qEnc}}_{\eta,3}(\mathsf{qMSK}, \widetilde{\mathbf{x}})$**:** It is the same as $\widehat{\mathsf{qEnc}}_{\eta,2}$ except that it defines vectors as follows:

$$\mathbf{r} = (r_i)_{i \in S_c} \leftarrow \mathbb{Z}_p^{S_c}, \quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, r_i + \cancel{x_{s_\eta} x_i})$$

$\widehat{\mathsf{qKeyGen}}_{\eta,3}(\mathsf{qMSK}, \widetilde{\mathbf{x}}, (S_k, \mathbf{c}))$**:** Let $\mathbf{r} = (r_i)_{i \in S_c}$ be the random vector chosen in $\widehat{\mathsf{qEnc}}_{\eta,3}$. It is the same as $\widehat{\mathsf{qKeyGen}}_{\eta,2}$ except that, if and only if $S_k \subseteq S_c$, it defines uSK as follows:

$$\mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [\widehat{d} + \sum_{i \in S_k} c_{s_\eta,i}(r_i \underline{- x_{s_\eta} x_i)}]_2))$$

Lemma 5.2 immediately follows from Lemmata 5.3 to 5.6. □

Lemma 5.3. *For* $\eta \in [s_{\mathsf{max}}]$*, we have* $\mathsf{H}_{\eta-1} \approx_c \widehat{\mathsf{H}}_{\eta,1}$ *if* iFE *and* uFE *are IND-(1-)partially hiding.*

Proof. First, we consider the case of $\eta \geq 2$. Let $\mathbf{b}_i^0, \widetilde{\mathbf{b}}_i^0$ be $\mathbf{b}_i, \widetilde{\mathbf{b}}_i$ defined in $\mathsf{H}_{\eta-1}$, i.e., Eq. (5.1), and $\mathbf{b}_i^1, \widetilde{\mathbf{b}}_i^1$ be $\mathbf{b}_i, \widetilde{\mathbf{b}}_i$ defined in $\widehat{\mathsf{H}}_{\eta,1}$, i.e., Eq. (5.3). Then, it is not hard to see that we have $\langle \mathbf{b}_i^0, \widetilde{\mathbf{b}}_j^0 \rangle = \langle \mathbf{b}_i^1, \widetilde{\mathbf{b}}_j^1 \rangle$ for all $i, j \in S_c$. Thus, we can reduce the indistinguishability between the 0-side and 1-side to partially-hiding security of iFE.

Let $\mathbf{d}_i^0$ be $\mathbf{d}_i$ defined in $\mathsf{H}_{\eta-1}$, i.e., Eq. (5.2), and $\mathbf{d}_i^1$ be $\mathbf{d}_i$ defined in $\widehat{\mathsf{H}}_{\eta,1}$, i.e., Eq. (5.4). Then, for $\ell \in [q_k]$ where q_k is the number of queries to the key generation oracle, we have

$$\sum_{i \in S_k^\ell} \langle \mathbf{d}_i^0, \widetilde{\mathbf{d}}_i^\ell \rangle + \widehat{d} = \sum_{i \in S_k^\ell} \langle \mathbf{d}_i^1, \widetilde{\mathbf{d}}_i^\ell \rangle + \widehat{d} + \sum_{i \in S_k^\ell} c_{s_\eta,i} \langle \mathbf{z}_{s_\eta}, \mathbf{a}_i \rangle \qquad \text{if } S_k^\ell \subseteq S_c$$

where $c_{s_\eta,i} = 0$ if $s_\eta \notin S_k$. Thus, we can reduce the indistinguishability between the 0-side and 1-side to the partially function-hiding property of uFE.

Next, we consider the case of $\eta = 1$, which can be similarly proven to the case of $\eta \geq 2$. A main difference is that we need to first change $\mathsf{uSlotEnc}(S_c, [\mathbf{d}]_1)$ in qEnc to $\mathsf{uEnc}(\mathsf{uMSK}, (S_c, [\mathbf{d}]_1, [0]_1))$, which are identically distributed by the slot-mode correctness of uFE. The remaining proof is almost the same as the case of $\eta \geq 2$. □

Lemma 5.4. *Let q_r be the maximum number of queries to the random oracle H in the security game. For all $\eta \in [s_{\mathsf{max}}]$, we have $\widehat{\mathsf{H}}_{\eta,1} \approx_c \widehat{\mathsf{H}}_{\eta,2}$ if the MDDH$_k$ assumption holds in $\mathbb{G}$.*

Proof. We can construct an adversary $\mathcal{B}$ against an MDDH$_k$ problem from a distinguisher $\mathcal{A}$ of the two hybrids as follows.

1. $\mathcal{B}$ obtains a $\mathcal{U}_{q_r,k}$-MDDH instance $(\mathbb{G}, [\mathbf{A}]_2, [\mathbf{k}_\delta]_2)$, where $\mathbf{A} \in \mathbb{Z}_p^{q_r \times k}$, $\mathbf{k}_0 = \mathbf{Az}$, $\mathbf{k}_1 \leftarrow \mathbb{Z}_p^{q_r}$.
2. $\mathcal{B}$ simulates the random oracle H as follows: when H is queried on $i \in [p]$ as the j-th fresh query to H, it returns $[\mathbf{a}_i]_2$ where $\mathbf{a}_i$ is the j-th row of $\mathbf{A}$. $\mathcal{B}$ also defines k_i as the j-th entry of $\mathbf{k}_\delta$.
3. $\mathcal{B}$ runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uSetup}(1^\lambda)$ and gives $\mathsf{qPK} = (\mathbb{G}, \mathsf{uPK})$ to $\mathcal{A}$. It sets $\mathsf{qMSK} = \mathsf{uMSK}$.
4. When $\mathcal{A}$ outputs $\widetilde{\mathbf{x}}$, $\mathcal{B}$ computes qCT in the same way as $\widehat{\mathsf{qEnc}}_{\eta,1}$ except that it defines $\widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, k_i + x_{s_\eta} x_i)$.
5. When $\mathcal{A}$ queries to the key generation oracle on $(S_k, \mathbf{c})$, $\mathcal{B}$ computes qSK in the same way as $\widehat{\mathsf{qKeyGen}}_{\eta,1}$ except that it computes uSK as $\mathsf{uSK} \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}]_2, [\widehat{d} + \sum_{i \in S_k} c_{s_\eta,i} k_i]_2))$ if $S_k \subseteq S_c$.
6. $\mathcal{B}$ outputs what $\mathcal{A}$ outputs.

It is not hard to see that $\mathcal{A}$'s view corresponds to $\widehat{\mathsf{H}}_{\eta,1}$ if $\delta = 0$ and $\widehat{\mathsf{H}}_{\eta,2}$ otherwise. □

Lemma 5.5. *For $\eta \in [s_{\mathsf{max}}]$, $\widehat{\mathsf{H}}_{\eta,2}$ and $\widehat{\mathsf{H}}_{\eta,3}$ are identically distributed.*

Proof. For $i \in S_c$, by implicitly defining $r_i = r'_i - x_{s_\eta} x_i$ where $r'_i \leftarrow \mathbb{Z}_p$, it is obvious that $\mathcal{A}$'s views in $\widehat{\mathsf{H}}_{\eta,2}$ and $\widehat{\mathsf{H}}_{\eta,3}$ are identical since the distribution of r_i is not changed from the original definition. □

Lemma 5.6. *For $\eta \in [s_{\mathsf{max}}]$, we have $\widehat{\mathsf{H}}_{\eta,3} \approx_c \mathsf{H}_\eta$ if iFE and uFE are IND-(1-)partially hiding and the MDDH$_k$ assumption holds in $\mathbb{G}$.*

This lemma can be proven similarly to lemmata 5.3 to 5.4.

5.3 Bounded Variable-Length Scheme without Random Oracles.

The scheme in Section 5.1 is easily modified into a bounded variable-length scheme that does not rely on random oracles. Note that the functionality of the scheme is the same as Definition 2.7 except that S_c and S_k is subsets of a fixed polynomial-sized set $[n']$ instead of $[p]$. The modification is simple: the setup algorithm randomly chooses $[\mathbf{a}_1]_2, \ldots, [\mathbf{a}_{n'}]_2$ from G_2^k and publish these. The encryption and key generation algorithms use them instead of computing by the hash function on the fly.

6 Functional Encryption for ABP ∘ UQF

In this section, we present our FE scheme for unbounded quadratic functions defined in Definition 2.8.

6.1 Partial Garbling Scheme for $\mathcal{F}_{n,n'}^{\mathsf{ABP}}$

We use the following partial garbling scheme for $\mathcal{F}_{n,n'}^{\mathsf{ABP}}$ [IW14] for the construction of our FE scheme.

Syntax. A partial garbling scheme for $\mathcal{F}_{n,n'}^{\mathsf{ABP}}$ consists of the four algorithms. Note that lgen and rec are deterministic algorithms while pgb and pgb^* are probabilistic algorithms.

$\mathsf{lgen}(f)$**:** It takes $f \in \mathcal{F}_{n,n'}^{\mathsf{ABP}}$ and outputs $\mathbf{L}_1, \ldots, \mathbf{L}_t \in \mathbb{Z}_p^{(n+1)\times(t-1)}$ where t depends on f.

$\mathsf{pgb}(f, \mathbf{u}, \mathbf{x}; \mathbf{t})$**:** Let $\mathbf{u}'^\top = (\mathbf{u}, 1)$. It takes $f \in \mathcal{F}_{n,n'}^{\mathsf{ABP}}, \mathbf{u} \in \mathbb{Z}_p^n, \mathbf{x} \in \mathbb{Z}_p^{n'}$, and a random tape $\mathbf{t} \in \mathbb{Z}_p^{t-1}$. It then outputs

$$(\mathbf{u}'^\top \mathbf{L}_1 \mathbf{t}, \ldots, \mathbf{u}'^\top \mathbf{L}_m \mathbf{t}, x_1 + \mathbf{u}'^\top \mathbf{L}_{m+1}\mathbf{t}, \ldots, x_{n'} + \mathbf{u}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $m = t - n'$ and $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

$\mathsf{pgb}^*(f, \mathbf{u}, \mu; \mathbf{t})$**:** It takes $\mu \in \mathbb{Z}_p$ and $f, \mathbf{u}, \mathbf{t}$ as above and outputs

$$(\mathbf{u}'^\top \mathbf{L}_1 \mathbf{t} + \mu, \mathbf{u}'^\top \mathbf{L}_2 \mathbf{t}, \ldots, \mathbf{u}'^\top \mathbf{L}_t \mathbf{t}) \in \mathbb{Z}_p^t$$

where $(\mathbf{L}_1, \ldots, \mathbf{L}_t) = \mathsf{lgen}(f)$.

$\mathsf{rec}(f, \mathbf{u})$**:** It takes $f, \mathbf{u} \in \mathbb{Z}_p^n$ and outputs $\mathbf{d}_{f,\mathbf{u}} \in \mathbb{Z}_p^t$.

The concrete description of $\mathsf{lgen}, \mathsf{rec}$ that satisfies the following properties is found in [AGW20, Appendix A]. We slightly modify the format of the output of lgen from [AGW20] for convenience in our construction, but note that they are essentially the same.

Correctness. The garbling scheme is correct if for all $f \in \mathcal{F}_{n,n'}^{\mathsf{ABP}}, \mathbf{u} \in \mathbb{Z}_p^n, \mathbf{x} \in \mathbb{Z}_p^{n'}, \mathbf{t} \in \mathbb{Z}_p^{t-1}$, we have

$$\langle \mathsf{pgb}(f, \mathbf{u}, \mathbf{x}; \mathbf{t}), \mathsf{rec}(f, \mathbf{u}) \rangle = \langle f(\mathbf{u}), \mathbf{x} \rangle.$$

Security. The garbling scheme is secure if for all $f \in \mathcal{F}_{n,n'}^{\mathsf{ABP}}, \mathbf{u} \in \mathbb{Z}_p^n, \mathbf{x} \in \mathbb{Z}_p^{n'}$, the following distributions are statistically close:

$$\mathsf{pgb}(f, \mathbf{u}, \mathbf{x}; \mathbf{t}) \quad \text{and} \quad \mathsf{pgb}^*(f, \mathbf{u}, \langle f(\mathbf{u}), \mathbf{x} \rangle; \mathbf{t})$$

where the random tape is chosen over $\mathbf{t} \leftarrow \mathbb{Z}_p^{t-1}$.

Linearlity. Observe that pgb and pgb^* are an affine functions in $\mathbf{t}$ and μ, respectively. This means that $\mathbf{t}$ in pgb and μ in pgb^* can be group elements of order p.

6.2 Construction

Let k be the parameter of the MDDH_k assumption, n be the input length of arithmetic branching programs in $\mathcal{F}_{n,\mathbb{G}}^{\mathsf{ABP}\circ\mathsf{UQF}}$. Let $\mathsf{uFE} = (\mathsf{uSetup}, \mathsf{uEnc}, \mathsf{uSlotEnc}, \mathsf{uKeyGen}, \mathsf{uDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}_{k(n+1),1,\mathbb{G}}^{\mathsf{UIP}}$ with slot-mode correctness for $e = [0]_1$. Let $(\mathsf{lgen}, \mathsf{pgb}, \mathsf{pgb}^*, \mathsf{rec})$ be a partial garbling scheme defined in the above. Let $H : [p] \to G_2^k$ be a hash function modeled as a random oracle. Then, our partially hiding FE scheme $\mathsf{aFE} = (\mathsf{aSetup}, \mathsf{aEnc}, \mathsf{aKeyGen}, \mathsf{aDec})$ for $\mathcal{F}_{n,\mathbb{G}}^{\mathsf{ABP}\circ\mathsf{UQF}}$ is constructed as follows.

$\mathsf{aSetup}(1^\lambda)$**:** It runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uKeyGen}(1^\lambda)$ outputs $(\mathsf{aPK}, \mathsf{aMSK}) = (\mathsf{uPK}, \mathsf{uMSK})$.

$\mathsf{aEnc}(\mathbf{u}, S_c, \mathbf{x} = (x_i)_{i\in S_c})$**:** First, it defines vectors as follows:

$$[\mathbf{a}_i]_2 = H(i),\ \mathbf{z}_i, \widetilde{\mathbf{z}} \leftarrow \mathbb{Z}_p^k,\ \mathbf{b}_i = (x_i, \mathbf{z}_i, 0),\ \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, 0)$$
$$\mathbf{d}_i = \begin{cases} (\mathbf{z}_i, 0^{kn}) & (i \in S_c) \\ (\mathbf{u}, 1) \otimes \widetilde{\mathbf{z}} & (i = p) \end{cases},\ \mathbf{d} = (\mathbf{d}_i)_{i\in S_c\cup\{p\}}. \tag{6.1}$$

Then, it outputs aCT as follows: let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}_{0,k+2,\mathbb{G}}^{\mathsf{IP}}$ with slot-mode correctness for $e = [0^{k+2}]_1$, or equivalently standard function-hiding IPFE scheme with the vector length being $k+2$.

$$(\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$$
$$\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{b}_i]_1),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{b}}_i]_2)$$
$$\mathsf{uCT} \leftarrow \mathsf{uSlotEnc}(S_c \cup \{p\}, [\mathbf{d}]_1),\ \mathsf{aCT} = (\mathbf{u}, \mathsf{iPK}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i\in S_c}, \mathsf{uCT}) \tag{6.2}$$

$\mathsf{aKeyGen}(\mathsf{aMSK}, (S_k, f \in \mathcal{F}_{n,|S_k|^2}^{\mathsf{ABP}}))$**:** Let $\phi : S_k^2 \to \{m+1, \ldots, t\}$ be the bijective function defined as $\phi(\mu, \nu) = m + (\mu - 1)|S_k| + \nu$ (see Section 6.1 for how to define m, t). It outputs aSK as follows: first it computes $\mathbf{L}_1, \ldots, \mathbf{L}_t \leftarrow \mathsf{lgen}(f)$ and chooses $\mathbf{T} \leftarrow \mathbb{Z}_p^{(t-1)\times k}$. For $j \in [m], \mu, \nu \in S_k$, it defines

$$\widetilde{\mathbf{d}}_{j,i} = \begin{cases} \mathbf{0} & (i \in S_k) \\ \mathsf{vec}(\mathbf{L}_j\mathbf{T}) & (i = p) \end{cases},\quad \widetilde{\mathbf{d}}_{\phi(\mu,\nu),i} = \begin{cases} \mathbf{0} & (i \in S_k\backslash\{\mu\}) \\ (\mathbf{a}_\nu, 0^{kn}) & (i = \mu) \\ \mathsf{vec}(\mathbf{L}_{\phi(\mu,\nu)}\mathbf{T}) & (i = p) \end{cases}$$

where $[\mathbf{a}_\nu]_2 = H(\nu)$. It then defines $\widetilde{\mathbf{d}}_j = (\widetilde{\mathbf{d}}_{j,i})_{i\in S_k\cup\{p\}}$ for $j \in [t]$. Finally it computes $\mathsf{uSK}_j \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k \cup \{p\}, [\widetilde{\mathbf{d}}_j]_2, [0]_2))$ for all $j \in [t]$, and sets $\mathsf{aSK} = (f, \{\mathsf{uSK}_j\}_{j\in[t]})$.

$\mathsf{aDec}(\mathsf{aCT}, \mathsf{aSK})$**:** Parse $\mathsf{aCT} = (\mathbf{u}, \mathsf{iPK}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i\in S_c}, \mathsf{uCT})$ and $\mathsf{aSK} = (f, \{\mathsf{uSK}_j\}_{j\in[t]})$. If $S_k \not\subseteq S_c$, it outputs $\perp$. Otherwise, it computes $\mathbf{d}_{f,\mathbf{u}} = \mathsf{rec}(f, \mathbf{u})$ and outputs $[\delta]_T$ as follows:

$$[\delta_0]_T = \sum_{i,j\in S_k} f_{i,j}(\mathbf{u})\mathsf{iDec}(\mathsf{iCT}_i, \mathsf{iSK}_j),\ [\delta_i]_T = \mathsf{uDec}(\mathsf{uCT}, \mathsf{uSK}_i)$$
$$[\delta]_T = [\delta_0 - \langle \mathbf{d}_{f,\mathbf{u}}, \boldsymbol{\delta}\rangle]_T$$

where $\boldsymbol{\delta} = (\delta_1, \ldots, \delta_t)$.

Correctness. Due to the correctness of $\mathsf{iFE}, \mathsf{uEF}$, we have

$$\delta_0 = \sum_{i,j\in S_k} (f_{i,j}(\mathbf{u})x_i x_j + f_{i,j}(\mathbf{u})\langle \mathbf{z}_i, \mathbf{a}_j\rangle),\ \boldsymbol{\delta} = \mathsf{pgb}(f, \mathbf{u}, (\langle \mathbf{z}_i, \mathbf{a}_j\rangle)_{i,j\in S_k}; \mathbf{T}\widetilde{\mathbf{z}})$$

Hence, we have $\langle \mathbf{d}_{f,\mathbf{u}}, \boldsymbol{\delta}\rangle = \sum_{i,j\in S_k} f_{i,j}(\mathbf{u})\langle \mathbf{z}_i, \mathbf{a}_j\rangle$ and thus $z = \sum_{i,j\in S_k} f_{i,j}(\mathbf{u})x_i x_j$, which follows from the correctness of the partial garbling scheme.

6.3 Security

For security, we have the following theorem.

Theorem 6.1. *If* iFE *is IND-partially hiding,* uFE *is IND-1-partially hiding, the partial garbling scheme is secure, and the MDDH$_k$ assumption holds in* $\mathbb{G}$*, then* aFE *is SIM-partially-hiding.*

Proof. First, we show our simulation algorithms. Note that our simulation algorithm takes a G_2 element instead of a G_T element, which follows [Wee20, GQ20].

$\mathsf{aSetup}^*(1^\lambda)$**:** It runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uKeyGen}(1^\lambda)$ outputs $(\mathsf{aPK}^*, \mathsf{aMSK}^*) = (\mathsf{uPK}, \mathsf{uMSK})$.

$\mathsf{aEnc}^*(\mathsf{aMSK}^*, \mathbf{u}, S_c)$**:** First, it defines vectors as follows:

$$[\mathbf{a}_i]_2 = H(i),\ \mathbf{z}_i, \widetilde{\mathbf{z}} \leftarrow \mathbb{Z}_p^k,\ \mathbf{b}_i = (0, \mathbf{z}_i, 0),\ \widetilde{\mathbf{b}}_i = (0, \mathbf{a}_i, 0)$$
$$\mathbf{d}_i = \mathbf{0},\ \mathbf{d} = (\mathbf{d}_i)_{i\in S_c\cup\{p\}}.$$

Then, it outputs aCT^* as follows: let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{IP}}_{0,k+2,\mathbb{G}}$ with slot-mode correctness for $e = [0^{k+2}]_1$.

$$(\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$$
$$\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{b}_i]_1),\ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{b}}_i]_2)$$
$$\mathsf{uCT} \leftarrow \mathsf{uEnc}(\mathsf{uMSK}, (S_c \cup \{p\}, [\mathbf{d}]_1, [1]_1)),\ \mathsf{aCT}^* = (\mathbf{u}, \mathsf{iPK}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i\in S_c}, \mathsf{uCT})$$

$\mathsf{aKeyGen}^*(\mathsf{aMSK}^*, (S_k, f, \mathbf{u}, S_c, [\alpha]_2 \text{ or } \perp))$**:** Let $\phi : S_k^2 \to \{m+1, \ldots, t\}$ be the bijective function defined as $\phi(\mu, \nu) = m + (\mu - 1)|S_k| + \nu$ (see Section 6.1 for

$\underline{\mathsf{H}_s}$	$\underline{\mathsf{H}_\eta}$
$\mathsf{aPK}, \mathsf{aMSK} \leftarrow \mathsf{aSetup}(1^\lambda)$	$\mathsf{aPK}, \mathsf{aMSK} \leftarrow \mathsf{aSetup}(1^\lambda)$
$\widetilde{\mathbf{x}} = (\mathbf{u}, S_c, \mathbf{x}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{aPK})$	$\widetilde{\mathbf{x}} = (\mathbf{u}, S_c, \mathbf{x}) \leftarrow \mathcal{A}(1^\lambda, \mathsf{aPK})$
$\mathsf{aCT} \leftarrow \mathsf{aEnc}(\widetilde{\mathbf{x}})$	$\mathsf{aCT} \leftarrow \widetilde{\mathsf{aEnc}}(\mathsf{aMSK}, \widetilde{\mathbf{x}})$
$\beta \leftarrow \mathcal{A}^{\mathsf{aKeyGen}(\mathsf{aMSK}, \cdot)}(\mathsf{aCT})$	$\beta \leftarrow \mathcal{A}^{\widetilde{\mathsf{aKeyGen}}_\eta(\mathsf{aMSK}, \widetilde{\mathbf{x}}, \cdot)}(\mathsf{aCT})$

Fig 4. Hybrids for aFE.

how to define m, t). It outputs aSK^* as follows: first it computes $\mathbf{L}_1, \ldots, \mathbf{L}_t \leftarrow \mathsf{lgen}(f)$ and chooses $\mathbf{T} \leftarrow \mathbb{Z}_p^{(t-1)\times k}$. For $j \in [m], \mu, \nu \in S_k$, it defines

$$\widetilde{\mathbf{d}}_{j,i} = \begin{cases} \mathbf{0} & (i \in S_k) \\ \mathsf{vec}(\mathbf{L}_i \mathbf{T}) & (i = p) \end{cases}, \quad \widetilde{\mathbf{d}}_{\phi(\mu,\nu),i} = \begin{cases} \mathbf{0} & (i \in S_k \backslash \{\mu\}) \\ (\mathbf{a}_\nu, 0^{kn}) & (i = \mu) \\ \mathsf{vec}(\mathbf{L}_{\phi(\mu,\nu)} \mathbf{T}) & (i = p) \end{cases}$$

where $[\mathbf{a}_\nu]_2 = H(\nu)$. It then defines $\widetilde{\mathbf{d}}_j = (\widetilde{\mathbf{d}}_{j,i})_{i \in S_k \cup \{p\}}$ for $j \in [t]$. If $S_k \subseteq S_c$, it chooses $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$ and defines

$$\boldsymbol{\delta} = (\delta_1, \ldots, \delta_t) = \mathsf{pgb}^*(f, \mathbf{u}, -\alpha + \sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) \langle \mathbf{z}_i, \mathbf{a}_j \rangle; \widetilde{\mathbf{t}})$$

where $\mathbf{z}_i$ for $i \in S_k$ is the random vectors generated in aEnc [10]. Finally it computes

$$\mathsf{uSK}_j \leftarrow \begin{cases} \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k \cup \{p\}, [\widetilde{\mathbf{d}}_j]_2, [\delta_j]_2)) & S_k \subseteq S_c \\ \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k \cup \{p\}, [\widetilde{\mathbf{d}}_j]_2, [0]_2)) & \text{otherwise} \end{cases}$$

for all $j \in [t]$, and sets $\mathsf{aSK}^* = (f, \{\mathsf{uSK}_j\}_{j \in [t]})$.

We prove Theorem 6.1 via a series of hybrid games $\mathsf{H}_1, \mathsf{H}_2, \mathsf{H}_3, \mathsf{H}_f$. We show that $\mathsf{H}_s \approx_c \mathsf{H}_1 \approx_c \mathsf{H}_2 \approx_c \mathsf{H}_3 \approx_c \mathsf{H}_f$, where H_s is the real game. H_η for $\eta \in \{1, 2, 3\}$ is defined as described in Fig 4, where aEnc and $\mathsf{aKeyGen}$ are replaced with $\widetilde{\mathsf{aEnc}}$ and $\widetilde{\mathsf{aKeyGen}}_\eta$. They work as follows.

$\widetilde{\mathsf{aEnc}}(\mathsf{aMSK}, \widetilde{\mathbf{x}})$**:** It defines vectors as follows:

$$\begin{aligned} &[\mathbf{a}_i]_2 = H(i), \ \mathbf{z}_i, \widetilde{\mathbf{z}} \leftarrow \mathbb{Z}_p^k, \ \mathbf{b}_i = (x_i, \mathbf{z}_i, 0), \ \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, 0) \\ &\mathbf{d}_i = \underline{\mathbf{0}}, \ \ \mathbf{d} = (\mathbf{d}_i)_{i \in S_c \cup \{p\}}. \end{aligned} \tag{6.3}$$

Then, it outputs aCT as follows: let $\mathsf{iFE} = (\mathsf{iSetup}, \mathsf{iEnc}, \mathsf{iSlotEnc}, \mathsf{iKeyGen}, \mathsf{iDec})$ be a partially hiding slotted FE scheme for $\mathcal{F}^{\mathsf{IP}}_{0,k+2,\mathbb{G}}$ with slot-mode correctness for $e = [0^{k+2}]_1$.

$(\mathsf{iPK}, \mathsf{iMSK}) \leftarrow \mathsf{iSetup}(1^\lambda)$

$\mathsf{iCT}_i \leftarrow \mathsf{iEnc}(\mathsf{iMSK}, [\mathbf{b}_i]_1), \ \mathsf{iSK}_i \leftarrow \mathsf{iKeyGen}(\mathsf{iMSK}, [\widetilde{\mathbf{b}}_i]_2)$

$\mathsf{uCT} \leftarrow \underline{\mathsf{u}}\mathsf{Enc}(\mathsf{uMSK}, (S_c, [\mathbf{d}]_1, [1]_1)), \ \mathsf{aCT} = (\mathbf{u}, \mathsf{iPK}, \{\mathsf{iCT}_i, \mathsf{iSK}_i\}_{i \in S_c \cup \{p\}}, \mathsf{uCT})$

[10] Sharing randomness between aEnc^* and $\mathsf{aKeyGen}^*$ does not violate the definition of SIM-security. Just including a key of a pseudorandom function in aMSK^* suffices.

$\widetilde{\mathsf{aKeyGen}}_1(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** This algorithm is the same as $\mathsf{aKeyGen}$ except the following: if and only if $S_k \subseteq S_c$, it computes

$$\boldsymbol{\delta} = (\delta_1, \ldots, \delta_t) = \mathsf{pgb}(f, \mathbf{u}, (\langle \mathbf{z}_i, \mathbf{a}_j \rangle)_{i,j \in S_k}; \mathbf{T}\widetilde{\mathbf{z}})$$

where $\mathbf{z}_i$ and $\widetilde{\mathbf{z}}$ are values chosen in $\widetilde{\mathsf{aEnc}}$ to compute aCT. Then it computes uSK_i as $\mathsf{uSK}_i \leftarrow \mathsf{uKeyGen}(\mathsf{uMSK}, (S_k, [\widetilde{\mathbf{d}}_i]_2, \underline{[\delta_i]_2}))$ for $i \in [t]$.

$\widetilde{\mathsf{aKeyGen}}_2(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** This algorithm is the same as $\widetilde{\mathsf{aKeyGen}}_1$ except that it additionally chooses $\widetilde{\mathbf{t}} \leftarrow \mathbb{Z}_p^{t-1}$ and defines $\boldsymbol{\delta}$ as

$$\boldsymbol{\delta} = (\delta_1, \ldots, \delta_t) = \mathsf{pgb}(f, \mathbf{u}, (\langle \mathbf{z}_i, \mathbf{a}_j \rangle)_{i,j \in S_k}; \underline{\widetilde{\mathbf{t}}}).$$

$\widetilde{\mathsf{aKeyGen}}_3(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** This algorithm is the same as $\widetilde{\mathsf{aKeyGen}}_2$ except that it defines $\boldsymbol{\delta}$ as

$$\boldsymbol{\delta} = (\delta_1, \ldots, \delta_t) = \underline{\mathsf{pgb}^*(f, \mathbf{u}, \sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) \langle \mathbf{z}_i, \mathbf{a}_j \rangle; \widetilde{\mathbf{t}})}. \tag{6.4}$$

H_f is the same as H_3 except that $\widetilde{\mathsf{aEnc}}$ sets $\underline{\mathbf{b}_i = (0, \mathbf{z}_i, 0)}$, $\widetilde{\mathbf{b}}_i = (0, \mathbf{a}_i, 0)$ instead of $\mathbf{b}_i = (x_i, \mathbf{z}_i, 0), \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, 0)$ in Eq. (6.3), and $\widetilde{\mathsf{aKeyGen}}_3$ sets

$$\boldsymbol{\delta} = (\delta_1, \ldots, \delta_t) = \mathsf{pgb}^*(f, \mathbf{u}, \sum_{i,j \in S_k} f_{i,j}(\mathbf{u})(\langle \mathbf{z}_i, \mathbf{a}_j \rangle - \underline{x_i x_j}); \widetilde{\mathbf{t}}).$$

instead of Eq. (6.4). Observe that the adversary's view in H_f is the same as that in the simulated game. Thanks to Lemmata 6.1 to 6.4, Theorem 6.1 holds. □

Lemma 6.1. $\mathsf{H}_s \approx_c \mathsf{H}_1$ *if* uFE *is IND-1-partially hiding.*

Proof. When generating the challenge ciphertext in H_s, uCT in the challenge ciphertext is generated as $\mathsf{uCT} \leftarrow \mathsf{uSlotEnc}(S_c \cup \{p\}, [\mathbf{d}]_1)$ as described in Eq. (6.2). Even if the way of generating uCT is changed as $\mathsf{uCT} \leftarrow \mathsf{uEnc}(\mathsf{uMSK}, (S_c \cup \{p\}, [\mathbf{d}]_1, [0]_1))$, the adversary's view is not changed due to the slot-mode correctness of uFE.

For $i \in S_c \cup \{p\}$, let $\mathbf{d}_i^0$ be $\mathbf{d}_i$ defined in H_s, i.e., Eq. (6.1), and $\mathbf{d}_i^1$ be $\mathbf{d}_i$ defined in H_1, i.e., Eq. (6.3). Then, for all $\ell \in [q_k], j \in [t^\ell]$, we have

$$\sum_{i \in S_k^\ell \cup \{p\}} \langle \mathbf{d}_i^0, \widetilde{\mathbf{d}}_{j,i}^\ell \rangle = \sum_{i \in S_k^\ell \cup \{p\}} \langle \mathbf{d}_i^1, \widetilde{\mathbf{d}}_{j,i}^\ell \rangle + \delta_j^\ell \quad \text{if } S_k^\ell \subseteq S_c$$

where $\widetilde{\mathbf{d}}_{j,i}^\ell$ is $\widetilde{\mathbf{d}}_{j,i}$ defined in the ℓ-th secret key query, and

$$\begin{aligned} \delta_j^\ell &= \mathsf{pgb}_j(f^\ell, \mathbf{u}, (\langle \mathbf{z}_i, \mathbf{a}_j \rangle)_{i,j \in S_k}; \mathbf{T}^\ell \widetilde{\mathbf{z}}) \\ &= \begin{cases} \mathbf{u}'^\top \mathbf{L}_j^\ell \mathbf{T}^\ell \widetilde{\mathbf{z}} & (j \in [m^\ell]) \\ \langle \mathbf{z}_\mu, \mathbf{a}_\nu \rangle + \mathbf{u}'^\top \mathbf{L}_j^\ell \mathbf{T}^\ell \widetilde{\mathbf{z}} & (j \in \{m^\ell + 1, \ldots, t^\ell\}) \end{cases} \end{aligned}$$

where $\phi(\mu, \nu) = j$. Thus, we can reduce the indistinguishability between the 0-side and 1-side, which corresponds to H_s and H_1, respectively, to the partially-hiding security of uFE. □

Lemma 6.2. $\mathsf{H}_1 \approx_c \mathsf{H}_2$ *if the MDDH$_k$ assumption holds in* $\mathbb{G}$.

Proof. We can construct an adversary $\mathcal{B}$ against an MDDH$_k$ problem from a distinguisher $\mathcal{A}$ of the two hybrids as follows.

1. Let $q_t = \sum_{\ell \in [q_k]} t^\ell$. $\mathcal{B}$ obtains a $\mathcal{U}_{q_t,k}$-MDDH instance $(\mathbb{G}, [\mathbf{A}]_2, [\mathbf{k}_\delta]_2)$, where $\mathbf{A} \in \mathbb{Z}_p^{q_t \times k}$, $\mathbf{k}_0 = \mathbf{A}\mathbf{z}$, $\mathbf{k}_1 \leftarrow \mathbb{Z}_p^{q_t}$.
2. $\mathcal{B}$ runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uSetup}(1^\lambda)$ and gives $\mathsf{aPK} = (\mathbb{G}, \mathsf{uPK})$ to $\mathcal{A}$. It sets $\mathsf{aMSK} = \mathsf{uMSK}$.
3. When $\mathcal{A}$ outputs $\widetilde{\mathbf{x}}$, $\mathcal{B}$ computes aCT in the same way as $\widetilde{\mathsf{aEnc}}$.
4. When $\mathcal{A}$ queries to the key generation oracle on (S_k^ℓ, f^ℓ) in the ℓ-th query, $\mathcal{B}$ computes aSK in the same way as $\widetilde{\mathsf{aKeyGen}}_{\eta,1}$ except that it computes
$$[\boldsymbol{\delta}^\ell]_2 = \mathsf{pgb}(f^\ell, \mathbf{u}, (\langle \mathbf{z}_i, \mathbf{a}_j \rangle)_{i,j \in S_k}; [\mathbf{k}^\ell]_2)$$
where $\mathbf{k}^\ell$ is the vector consisting of the $\sum_{\ell' \in [\ell-1]} t^{\ell'} + 1$ to $\sum_{\ell' \in [\ell]} t^{\ell'}$ entries of $\mathbf{k}_\delta$. Note that since pgb is affine in $[\mathbf{k}^\ell]_2$, $\mathcal{B}$ can efficiently compute $[\boldsymbol{\delta}^\ell]_2$.
5. $\mathcal{B}$ outputs what $\mathcal{A}$ outputs.

It is not hard to see that $\mathcal{A}$'s view corresponds to H_1 if $\delta = 0$ and H_2 otherwise. □

Lemma 6.3. $\mathsf{H}_2 \approx_s \mathsf{H}_3$.

Lemma 6.3 directly follows from the security of the partial garbling scheme.

Lemma 6.4. $\mathsf{H}_3 \approx_c \mathsf{H}_f$ *if* iFE *is partially hiding and the MDDH$_k$ assumption holds in* $\mathbb{G}$.

Proof. The proof of Lemma 6.4 is similar to that of Theorem 5.1. We define intermediate hybrids $\widehat{\mathsf{H}}_\eta$ for $\eta \in [s_{\max}]$ where $s_{\max}$ is the maximum size of the challenge index set S_c. We show that $\mathsf{H}_3 \approx_c \widehat{\mathsf{H}}_1 \approx_c \cdots \approx_c \widehat{\mathsf{H}}_{s_{\max}} \approx_c \mathsf{H}_f$. $\widehat{\mathsf{H}}_i$ for $\eta \in [s_{\max}]$ is the same as H_3 except that $\widetilde{\mathsf{aEnc}}$ and $\widetilde{\mathsf{aKeyGen}}_3$ are replaced with $\widehat{\mathsf{aEnc}}_\eta$ and $\widehat{\mathsf{aKeyGen}}_\eta$. They work as follows for $\eta \in [s_{\mathsf{max}}]$.

$\widehat{\mathsf{aEnc}}_\eta(\mathsf{aMSK}, \widetilde{\mathbf{x}})$**:** Let $S_c = (s_1, \ldots, s_{|S_c|})$. This algorithm is the same as $\widetilde{\mathsf{aEnc}}$ except that it sets $\mathbf{b}_i$ and $\widetilde{\mathbf{b}}_i$ in Eq. (6.3) as

$$\mathbf{b}_i = \begin{cases} (\underline{0}, \ \mathbf{z}_i, 0) & (i \le s_\eta) \\ (x_i, \mathbf{z}_i, 0) & (i > s_\eta) \end{cases}, \quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, 0) \tag{6.5}$$

$\widehat{\mathsf{aKeyGen}}_\eta(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** Let $S_{c,\eta} = (s_1, \ldots, s_\eta)$ where s_i is the i-th element of the challenge index set S_c. This algorithm is the same as $\widetilde{\mathsf{aKeyGen}}_3$ except that it defines $\boldsymbol{\delta}$ as
$$\boldsymbol{\delta} = \mathsf{pgb}^*(f, \mathbf{u}, \sum_{i,j \in S_k} f_{i,j}(\mathbf{u}) \langle \mathbf{z}_i, \mathbf{a}_j \rangle - \underline{\sum_{\substack{i \in S_{c,\eta} \cap S_k \\ j \in S_k}} f_{i,j}(\mathbf{u}) x_i x_j}; \widetilde{\mathbf{t}}).$$

Thanks to Lemmata 6.5 and 6.6, Lemma 6.4 holds. ◻

Lemma 6.5. $\widehat{\mathsf{H}}_{s_{\mathsf{max}}} \approx_c \mathsf{H}_f$ *if* iFE *is IND-partially hiding.*

Proof. For all $i \in [|S_c|]$, let $\mathbf{b}_i^0$ and $\widetilde{\mathbf{b}}_i^0$ be $\mathbf{b}_i$ and $\widetilde{\mathbf{b}}_i$ defined in $\widehat{\mathsf{H}}_{s_{\mathsf{max}}}$. Similarly, let $\mathbf{b}_i^1$ and $\widetilde{\mathbf{b}}_i^1$ be $\mathbf{b}_i$ and $\widetilde{\mathbf{b}}_i$ defined in H_f. Then, it is not hard to see that $\langle \mathbf{b}_i^0, \widetilde{\mathbf{b}}_j^0 \rangle = \langle \mathbf{b}_i^1, \widetilde{\mathbf{b}}_j^1 \rangle$ for all $i, j \in [|S_c|]$. Hence, the difference between $\widehat{\mathsf{H}}_{s_{\mathsf{max}}}$ and H_f can be reduced to partially hiding security of iFE. ◻

Lemma 6.6. *Let* $\mathsf{H}_3 = \widehat{\mathsf{H}}_0$. *For* $\eta \in [s_{\mathsf{max}}]$, *we have* $\widehat{\mathsf{H}}_{\eta-1} \approx_c \widehat{\mathsf{H}}_\eta$ *if* iFE *is IND-partially hiding and the MDDH$_k$ assumption holds in* $\mathbb{G}$.

Proof. We define intermediate hybrids $\widehat{\mathsf{H}}_{\eta,1}, \widehat{\mathsf{H}}_{\eta,2}, \widehat{\mathsf{H}}_{\eta,3}$ and prove that $\widehat{\mathsf{H}}_{\eta-1} \approx_c \widehat{\mathsf{H}}_{\eta,1} \approx_c \widehat{\mathsf{H}}_{\eta,2} \approx_c \widehat{\mathsf{H}}_{\eta,3} \approx_c \widehat{\mathsf{H}}_\eta$. $\widehat{\mathsf{H}}_{\eta,i}$ for $i \in \{1,2,3\}$ is the same as $\widehat{\mathsf{H}}_{\eta-1}$ except that $\widehat{\mathsf{aEnc}}_{\eta-1}, \widehat{\mathsf{aKeyGen}}_{\eta-1}$ are replaced by $\widehat{\mathsf{aEnc}}_{\eta,i}, \widehat{\mathsf{aKeyGen}}_{\eta,i}$, respectively, which work as follows:

$\widehat{\mathsf{aEnc}}_{\eta,1}(\mathsf{aMSK}, \widetilde{\mathbf{x}})$**:** It is the same as $\widehat{\mathsf{aEnc}}_{\eta-1}$ except that it defines vectors as follows:

$$\mathbf{b}_i = \begin{cases} (0,\ \mathbf{z}_i, 0) & (i < s_\eta) \\ \underline{(0,\ \mathbf{0},\ 1)} & (i = s_\eta) \\ (x_i, \mathbf{z}_i, 0) & (i > s_\eta) \end{cases}, \quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, \underline{\langle \mathbf{z}_{s_\eta}, \mathbf{a}_i \rangle + x_{s_\eta} x_i}) \tag{6.6}$$

$\widehat{\mathsf{aKeyGen}}_{\eta,1}(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** It is the same as $\widehat{\mathsf{aKeyGen}}_{\eta-1}$.

$\widehat{\mathsf{aEnc}}_{\eta,2}(\mathsf{aMSK}, \widetilde{\mathbf{x}})$**:** It is the same as $\widehat{\mathsf{aEnc}}_{\eta,1}$ except that it defines vectors as follows:

$$\underline{\mathbf{r} = (r_i)_{i \in S_c} \leftarrow \mathbb{Z}_p^{S_c}}, \quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, \underline{r_i} + x_{s_\eta} x_i)$$

$\widehat{\mathsf{aKeyGen}}_{\eta,2}(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** Let $\mathbf{r} = (r_i)_{i \in S_c}$ be the random vector chosen in $\widehat{\mathsf{aEnc}}_{\eta,2}$. It is the same as $\widehat{\mathsf{aKeyGen}}_{\eta,1}$ except that it defines $\boldsymbol{\delta}$ as follows:

$$\boldsymbol{\delta} = \mathsf{pgb}^*(f, \mathbf{u}, \underline{\sum_{\substack{i \in S_k \setminus \{s_\eta\}, \\ j \in S_k}} f_{i,j}(\mathbf{u}) \langle \mathbf{z}_i, \mathbf{a}_j \rangle + \sum_{j \in S_k} f_{s_\eta, j}(\mathbf{u}) r_j} - \sum_{\substack{i \in S_{c,\eta-1} \cap S_k \\ j \in S_k}} f_{i,j}(\mathbf{u}) x_i x_j; \widetilde{\mathbf{t}})$$

where $f_{s_\eta, j}(\mathbf{u}) = 0$ if $s_\eta \notin S_k$.

$\widehat{\mathsf{aEnc}}_{\eta,3}(\mathsf{aMSK}, \widetilde{\mathbf{x}})$**:** It is the same as $\widehat{\mathsf{aEnc}}_{\eta,2}$ except that it defines vectors as follows:

$$\mathbf{r} = (r_i)_{i \in S_c} \leftarrow \mathbb{Z}_p^{S_c}, \quad \widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, r_i \cancel{+ x_{s_\eta} x_i})$$

$\widehat{\mathsf{aKeyGen}}_{\eta,3}(\mathsf{aMSK}, \widetilde{\mathbf{x}}, (S_k, f))$**:** Let $\mathbf{r} = (r_i)_{i \in S_c}$ be the random vector chosen in $\widehat{\mathsf{aEnc}}_{\eta,3}$. It is the same as $\widehat{\mathsf{aKeyGen}}_{\eta,2}$ except that it defines $\boldsymbol{\delta}$ as follows:

$$\boldsymbol{\delta} = \mathsf{pgb}^*(f, \mathbf{u}, \sum_{\substack{i \in S_k \setminus \{s_\eta\}, \\ j \in S_k}} f_{i,j}(\mathbf{u}) \langle \mathbf{z}_i, \mathbf{a}_j \rangle + \sum_{j \in S_k} f_{s_\eta, j}(\mathbf{u}) r_j - \sum_{\underline{\substack{i \in S_{c,\eta} \cap S_k \\ j \in S_k}}} f_{i,j}(\mathbf{u}) x_i x_j; \widetilde{\mathbf{t}}).$$

Lemma 6.6 immediately follows from Lemmata 6.7 to 6.10. $\square$

Lemma 6.7. *For $\eta \in [s_{\mathsf{max}}]$, we have $\mathsf{H}_{\eta-1} \approx_c \widehat{\mathsf{H}}_{\eta,1}$ if iFE is IND-partially hiding.*

Proof. Let $\mathbf{b}_i^0, \widetilde{\mathbf{b}}_i^0$ be $\mathbf{b}_i, \widetilde{\mathbf{b}}_i$ defined in $\mathsf{H}_{\eta-1}$, i.e., Eq. (6.5) for $\eta - 1$, and $\mathbf{b}_i^1, \widetilde{\mathbf{b}}_i^1$ be $\mathbf{b}_i, \widetilde{\mathbf{b}}_i$ defined in $\widehat{\mathsf{H}}_{\eta,1}$, i.e., Eq. (6.6). Then, it is not hard to see that we have $\langle \mathbf{b}_i^0, \widetilde{\mathbf{b}}_j^0 \rangle = \langle \mathbf{b}_i^1, \widetilde{\mathbf{b}}_j^1 \rangle$ for all $i, j \in S_c$. Thus, we can reduce the indistinguishability between the 0-side and 1-side to partially-hiding security of iFE. $\square$

Lemma 6.8. *Let q_r be the maximum number of queries to the random oracle H in the security game. For all $\eta \in [s_{\mathsf{max}}]$, we have $\widehat{\mathsf{H}}_{\eta,1} \approx_c \widehat{\mathsf{H}}_{\eta,2}$ if the MDDH$_k$ assumption holds in $\mathbb{G}$.*

Proof. We describe the reduction $\mathcal{B}$.

1. $\mathcal{B}$ obtains a $\mathcal{U}_{q_r,k}$-MDDH instance $(\mathbb{G}, [\mathbf{A}]_2, [\mathbf{k}_\delta]_2)$, where $\mathbf{A} \in \mathbb{Z}_p^{q_r \times k}$, $\mathbf{k}_0 = \mathbf{A}\mathbf{z}$, $\mathbf{k}_1 \leftarrow \mathbb{Z}_p^{q_r}$.
2. $\mathcal{B}$ simulates the random oracle H as follows: when H is queried on $i \in [p]$ as the j-th fresh query to H, it returns $[\mathbf{a}_i]_2$ where $\mathbf{a}_i$ is the j-th row of $\mathbf{A}$. $\mathcal{B}$ also defines k_i as the j-th entry of $\mathbf{k}_\delta$.
3. $\mathcal{B}$ runs $(\mathsf{uPK}, \mathsf{uMSK}) \leftarrow \mathsf{uSetup}(1^\lambda)$ and gives $\mathsf{aPK} = (\mathbb{G}, \mathsf{uPK})$ to $\mathcal{A}$. It sets $\mathsf{aMSK} = \mathsf{uMSK}$.
4. When $\mathcal{A}$ outputs $\widetilde{\mathbf{x}}$, $\mathcal{B}$ computes aCT in the same way as $\widehat{\mathsf{aEnc}}_{\eta,1}$ except that it defines $\widetilde{\mathbf{b}}_i = (x_i, \mathbf{a}_i, k_i + x_{s_\eta} x_i)$.
5. When $\mathcal{A}$ queries to the key generation oracle on (S_k, f), $\mathcal{B}$ computes aSK in the same way as $\widehat{\mathsf{aKeyGen}}_{\eta,1}$ except that it computes $[\boldsymbol{\delta}]_2$ as
$$[\boldsymbol{\delta}]_2 = \mathsf{pgb}^*(f, \mathbf{u}, \left[\sum_{i \in S_k \setminus \{s_\eta\}, j \in S_k} f_{i,j}(\mathbf{u}) \langle \mathbf{z}_i, \mathbf{a}_j \rangle + \sum_{j \in S_k} f_{s_\eta,j}(\mathbf{u}) k_i - \sum_{i \in S_{c,\eta-1} \cap S_k, j \in S_k} f_{i,j}(\mathbf{u}) x_i x_j \right]_2 ; \widetilde{\mathbf{t}})$$
Note that pgb^* is affine in the third input and thus $[\boldsymbol{\delta}]_2$ can be computed efficiently.
6. $\mathcal{B}$ outputs what $\mathcal{A}$ outputs.

It is not hard to see that $\mathcal{A}$'s view corresponds to $\widehat{\mathsf{H}}_{\eta,1}$ if $\delta = 0$ and $\widehat{\mathsf{H}}_{\eta,2}$ otherwise. $\square$

Lemma 6.9. *For $\eta \in [s_{\mathsf{max}}]$, $\widehat{\mathsf{H}}_{\eta,2}$ and $\widehat{\mathsf{H}}_{\eta,3}$ are identically distributed.*

Proof. For $i \in S_c$, by implicitly defining $r_i = r_i' - x_{s_\eta} x_i$ where $r_i' \leftarrow \mathbb{Z}_p$, it is obvious that $\mathcal{A}$'s views in $\widehat{\mathsf{H}}_{\eta,2}$ and $\widehat{\mathsf{H}}_{\eta,3}$ are identical since the distribution of r_i is not changed from the original definition. $\square$

Lemma 6.10. *For $\eta \in [s_{\mathsf{max}}]$, we have $\widehat{\mathsf{H}}_{\eta,3} \approx_c \mathsf{H}_\eta$ if iFE and uFE are IND-(1-)partially hiding and the MDDH$_k$ assumption holds in $\mathbb{G}$.*

This lemma can be proven similarly to lemmata 6.7 to 6.8.

References

ABDP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 733–751. Springer, Heidelberg, March / April 2015.

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

AGT21. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-input quadratic functional encryption from pairings. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 208–238, Virtual Event, August 2021. Springer, Heidelberg.

AGW20. Michel Abdalla, Junqing Gong, and Hoeteck Wee. Functional encryption for attribute-weighted sums from k-Lin. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part I*, volume 12170 of *LNCS*, pages 685–716. Springer, Heidelberg, August 2020.

AJL$^+$19. Prabhanjan Ananth, Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: New paradigms via low degree weak pseudorandomness and security amplification. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 284–332. Springer, Heidelberg, August 2019.

AJS18. Prabhanjan Ananth, Aayush Jain, and Amit Sahai. Indistinguishability obfuscation without multilinear maps: iO from LWE, bilinear maps, and weak pseudorandomness. Cryptology ePrint Archive, Report 2018/615, 2018. https://eprint.iacr.org/2018/615.

AS16. Prabhanjan Vijendra Ananth and Amit Sahai. Functional encryption for turing machines. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 125–153. Springer, Heidelberg, January 2016.

BCFG17. Carmen Elisabetta Zaira Baltico, Dario Catalano, Dario Fiore, and Romain Gay. Practical functional encryption for quadratic functions with applications to predicate encryption. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 67–98. Springer, Heidelberg, August 2017.

BCP14. Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 52–73. Springer, Heidelberg, February 2014.

BJK15. Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part I*, volume 9452 of *LNCS*, pages 470–491. Springer, Heidelberg, November / December 2015.

BS15. Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 306–324. Springer, Heidelberg, March 2015.

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

DP19. Edouard Dufour Sans and David Pointcheval. Unbounded inner-product functional encryption with succinct keys. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19*, volume 11464 of *LNCS*, pages 426–441. Springer, Heidelberg, June 2019.

DP21. Pratish Datta and Tapas Pal. (Compact) adaptively secure FE for attribute-weighted sums from k-lin. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part IV*, volume 13093 of *LNCS*, pages 434–467. Springer, Heidelberg, December 2021.

EHK$^+$17. Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Luis Villar. An algebraic framework for Diffie-Hellman assumptions. *Journal of Cryptology*, 30(1):242–288, January 2017.

Gay20. Romain Gay. A new paradigm for public-key functional encryption for degree-2 polynomials. In Aggelos Kiayias, Markulf Kohlweiss, Petros Wallden, and Vassilis Zikas, editors, *PKC 2020, Part I*, volume 12110 of *LNCS*, pages 95–120. Springer, Heidelberg, May 2020.

GGH13a. Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 1–17. Springer, Heidelberg, May 2013.

GGH$^+$13b. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GGHZ16. Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 480–511. Springer, Heidelberg, January 2016.

GJLS21. Romain Gay, Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from simple-to-state hard problems: New assumptions, new techniques, and simplification. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part III*, volume 12698 of *LNCS*, pages 97–126. Springer, Heidelberg, October 2021.

GQ20. Junqing Gong and Haifeng Qian. Simple and efficient FE for quadratic functions. Cryptology ePrint Archive, Report 2020/1026, 2020. https://eprint.iacr.org/2020/1026.

IPS15. Yuval Ishai, Omkant Pandey, and Amit Sahai. Public-coin differing-inputs obfuscation and its applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part II*, volume 9015 of *LNCS*, pages 668–697. Springer, Heidelberg, March 2015.

IW14. Yuval Ishai and Hoeteck Wee. Partial garbling schemes and their applications. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *ICALP 2014, Part I*, volume 8572 of *LNCS*, pages 650–662. Springer, Heidelberg, July 2014.

JLMS19. Aayush Jain, Huijia Lin, Christian Matt, and Amit Sahai. How to leverage hardness of constant-degree expanding polynomials overa $\mathbb{R}$ to build $i\mathcal{O}$. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 251–281. Springer, Heidelberg, May 2019.

Lin17. Huijia Lin. Indistinguishability obfuscation from SXDH on 5-linear maps and locality-5 PRGs. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part I*, volume 10401 of *LNCS*, pages 599–629. Springer, Heidelberg, August 2017.

LL20. Huijia Lin and Ji Luo. Compact adaptively secure ABE from k-Lin: Beyond NC^1 and towards NL. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 247–277. Springer, Heidelberg, May 2020.

LV16. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from DDH-like assumptions on constant-degree graded encodings. In Irit Dinur, editor, *57th FOCS*, pages 11–20. IEEE Computer Society Press, October 2016.

O'N10. Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. https://eprint.iacr.org/2010/556.

OT10. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 191–208. Springer, Heidelberg, August 2010.

RPB$^+$19. Théo Ryffel, David Pointcheval, Francis R. Bach, Edouard Dufour-Sans, and Romain Gay. Partially encrypted deep learning using functional encryption. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *NeurIPS 2019*, pages 4519–4530, 2019.

TT18. Junichi Tomida and Katsuyuki Takashima. Unbounded inner product functional encryption from bilinear maps. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, volume 11273 of *LNCS*, pages 609–639. Springer, Heidelberg, December 2018.

Wee20. Hoeteck Wee. Functional encryption for quadratic functions from k-lin, revisited. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020, Part I*, volume 12550 of *LNCS*, pages 210–228. Springer, Heidelberg, November 2020.