Two-Input Functional Encryption for Inner Products from Bilinear Maps

Kwangsu Lee* Dong Hoon Lee†

Abstract

Functional encryption is a new paradigm of public-key encryption that allows a user to compute $f(x)$ on encrypted data $CT(x)$ with a private key SK_f to finely control the revealed information. Multi-input functional encryption is an important extension of (single-input) functional encryption that allows the computation $f(x_1,\ldots,x_n)$ on multiple ciphertexts $CT(x_1),\ldots,CT(x_n)$ with a private key SK_f. Although multi-input functional encryption has many interesting applications like running SQL queries on encrypted database and computation on encrypted stream, current candidates are not yet practical since many of them are built on indistinguishability obfuscation. To solve this unsatisfactory situation, we show that practical two-input functional encryption schemes for inner products can be built based on bilinear maps. In this paper, we first propose a two-input functional encryption scheme for inner products in composite-order bilinear groups and prove its selective IND-security under simple assumptions. Next, we propose a two-client functional encryption scheme for inner products where each ciphertext can be associated with a time period and prove its selective IND-security. Furthermore, we show that our two-input functional encryption schemes in composite-order bilinear groups can be converted into schemes in prime-order asymmetric bilinear groups by using the asymmetric property of asymmetric bilinear groups.

Keywords: Functional encryption, Multi-input functional encryption, Inner product, Bilinear maps.

*Sejong University, Seoul, Korea. Email: kwangsu@sejong.ac.kr.
†Korea University, Seoul, Korea. Email: donghlee@korea.ac.kr.

1 Introduction

Functional encryption (FE) is a new paradigm of public-key encryption (PKE) [15, 16]. In traditional PKE, a receiver can obtain the entire message x from a ciphertext $CT(x)$ or nothing depending on whether he has a proper private key SK or not. In FE, a receiver can obtain $f(x)$ from a ciphertext $CT(x)$ if he has a private key SK_f for a function $f(\cdot)$ that is received from a trusted center. Note that the user can obtain nothing about x except $f(x)$. That is, FE allows computation $f(x)$ on encrypted data $CT(x)$ by allowing a user to calculate $f(x)$ on the message x through the decryption process if he has a private key SK_f. In terms of computation on encrypted data, one may think that FE is similar to fully homomorphic encryption (FHE) [23]. However, FE is quite different from FHE since a user cannot directly obtain the result of computation from FHE. That is, a ciphertext $CT(f(x_1,x_2))$ that is the encryption of a result is just obtained from two ciphertexts $CT(x_1)$, $CT(x_2)$, and a function f in FHE, whereas a result $f(x)$ is directly derived from a ciphertext $CT(x)$ and a private key SK_f in FE.

Multi-input functional encryption (MI-FE) is an extended notion of (single-input) FE that allows the decryption algorithm of MI-FE to take multiple ciphertexts as input [24]. In MI-FE, a sender can create multiple ciphertexts $CT(x_1), CT(x_2), \ldots, CT(x_n)$ independently, and then a receiver who has a private key SK_f received from a trusted center can computes $f(x_1,x_2,\ldots,x_n)$ from the multiple ciphertexts. In terms of computation on encrypted data, MI-FE is the proper notion for computation on encrypted data since it can handle multiple ciphertexts at once, whereas (single-input) FE is a weaker one since it handles only single ciphertext at once. MI-FE has many interesting applications like running SQL queries on encrypted database, computation on encrypted stream, and order-revealing encryption for database [14,24]. Additionally, MI-FE is a very powerful tool since it can be converted into an indistinguishability obfuscator [8, 24].

Although FE is a very powerful concept, it is not easy to construct an FE scheme. Many FE schemes were built on heavy cryptographic tools like garbling circuits with universal circuits, indistinguishability obfuscators (iO), and multilinear maps [21, 22, 26]. The situation of MI-FE is worse than that of FE since MI-FE schemes only can be built from iO. Although there are some generic conversion methods that can convert a single-input FE scheme into an MI-FE scheme [8, 19], these conversions are not efficient yet since the underlying FE scheme should support polynomial-size circuits. Another approach to build a practical FE scheme is to restrict the family of supported functions. Recently, FE schemes for inner products were proposed based on PKE schemes with some special properties [2, 3, 6]. Although the functionality of inner products is restrictive than that of all circuits, it is still attractive in practical scenarios like descriptive statistics. In this paper, we ask the following natural question for the practical construction of MI-FE:

"Can we build a practical two-input functional encryption scheme for inner products?"

1.1 Our Contributions

In this paper, we give an affirmative answer to the above question. That is, we show that it is possible to build a practical two-input functional encryption scheme for inner products by relying on bilinear maps. The following is the summary of our results:

Two-Input Functional Encryption. We first propose a two-input functional encryption (TI-FE) scheme that supports the functionality of inner products in composite-order bilinear maps and prove its security (selective IND-security). In TI-FE for inner products, a user who has an encryption key EK_j can create a ciphertext CT_j for a vector $\vec{x}_j$ where $j \in \{1,2\}$, and a receiver who has a private key $SK_{\vec{y}}$ for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ can compute $\langle \vec{x}, \vec{y} \rangle$ from two ciphertexts CT_1, CT_2 where $\vec{x} = (\vec{x}_1, \vec{x}_2)$. The basic idea of our TI-FE schemes is applying bilinear maps to the FE scheme of Abdalla et al. [2]. The main obstacle of constructing

a TI-FE scheme is to handle two independent random values in ciphertexts. To solve this problem, we aggregate these random values by applying the pairing operation. For the security proof, we use simple complexity assumptions, named the composite 2-party Diffie-Hellman (C2DH) assumption and the 3-party Diffie-Hellman (3DH) assumption, in composite-order bilinear groups. The detailed design principle of our TI-FE scheme is given in the later part of this section. Our TI-FE scheme is based on the FE scheme of Abdalla et al. [2] and it is selectively IND-secure under the C2DH and 3DH assumptions.

Two-Client Functional Encryption. Two-client functional encryption (TC-FE) is an interesting variant of TI-FE where two clients have their own encryption keys and a ciphertext is additionally associated with a time period T. That is, a client who has an encryption key EK_j can create a ciphertext $CT_{j,T}$ for a message $\vec{x}_j$ by associating the ciphertext with a time period T. If the time periods of two ciphertexts are equal, then a receiver who has a private key $SK_{\vec{y}}$ can computes $\langle \vec{x}, \vec{y} \rangle$ from two ciphertexts $CT_{1,T}, CT_{2,T}$. TC-FE can be viewed as a specific type of multi-client functional encryption (MC-FE) introduced by Goldwasser et al. [24]. To construct a TC-FE scheme, we modify our TI-FE scheme by incorporating the structure of identity-based encryption to deal with the time period. Our TC-FE scheme is also selectively IND-secure under the C2DH and 3DH assumptions.

Two-Input FE in Prime-Order Groups. Finally, we show that our TI-FE and TC-FE schemes in composite-order bilinear groups can be also built in prime-order asymmetric bilinear groups. The main idea of using prime-order asymmetric bilinear groups instead of composite-order bilinear groups is to use the asymmetry of pairing operations in asymmetric bilinear groups. That is, if we set the first ciphertext CT_1 in the group $\mathbb{G}$ and the second ciphertext CT_2 in another group $\hat{\mathbb{G}}$, then a simple distinguishing attack that uses a pairing operation can be thwarted since the decision Diffie-Hellman (DDH) assumption still hold in both groups of asymmetric bilinear groups. To prove the security of our schemes in asymmetric bilinear groups, we use the SXDH assumption and the asymmetric version of the 3DH assumptions.

1.2 Our Technique

An efficient FE scheme for inner products (FE-IP) was proposed by Abdalla et al. [2]. They showed that an FE-IP scheme can be built from the ElGamal PKE scheme by reusing the randomness of ciphertexts and proved its security under the DDH assumption. In their FE-IP scheme, a ciphertext is $CT = (g^t, \{g^{x_i} w_i^t\}_{i=1}^n)$ for a vector $\vec{x} = (x_1, \ldots, x_n)$ and a private key is $SK_y = \sum_{i=1}^n \omega_i y_i$ for a vector $\vec{y} = (y_1, \ldots, y_n)$ where $w_i = g^{\omega_i}$. To decrypt the ciphertext, it first obtains $g^{\langle \vec{x}, \vec{y} \rangle}$ by computing $\prod_{i=1}^n (g^{x_i} w_i^t)^{y_i} \cdot (g^t)^{-SK_y}$, and then it obtains $\langle \vec{x}, \vec{y} \rangle$ by solving the discrete logarithm of $g^{\langle \vec{x}, \vec{y} \rangle}$ if $\langle \vec{x}, \vec{y} \rangle$ is within a small range. The selective IND-security of this FE-IP scheme can be proven under the DDH assumption since the number of private keys is (essentially) limited.

To extend the single-input FE scheme for inner products to a two-input FE (TI-FE) scheme for inner products, we design a TI-FE scheme in bilinear groups. One problem of using bilinear groups is that the structure of the ElGamal PKE scheme cannot be directly used since the DDH assumption does not hold in bilinear groups. Another problem of designing a TI-FE scheme is that it is relatively hard to handle two independent random values of two ciphertexts in the decryption algorithm. Note that the FE-IP scheme of Abdalla et al. [2] does not have this problem since each components of the ciphertext shares the same random value.

To solve the first problem of the DDH assumption in bilinear groups, we use composite-order bilinear groups and define a new complexity assumption that has a structural similarity with the DDH assumption. That is, we set a bilinear group of composite order $N = p_1 p_2$ and form a ciphertext $CT = (g^t Q_1, \{g^{x_i} w_i^t Q_{2,i}\}_{i=1}^n)$ where $g, w_i \in \mathbb{G}_{p_1}$ and $Q, \{Q_{2,i}\} \in \mathbb{G}_{p_2}$. In this case, this scheme does not work in $\mathbb{G}$

because of the random elements $Q_1, \{Q_{2,i}\}$, but it works well in $\mathbb{G}_T$ since the additional random elements are removed by the pairing operation. Thus, it is still possible to build an FE-IP scheme that is similar to the scheme of Abdalla et al. in composite-order bilinear groups although it is relatively inefficient.

To solve the second problem of handling two independent random values in ciphertexts, we aggregate these random values by using the pairing operation instead of removing these random values in the decryption process. Let $CT_1 = (g^{t_1}, \{g^{x_{1,i}} w_{1,i}^{t_1}\})$ with a random t_1 and $CT_2 = (g^{t_2}, \{g^{x_{2,i}} w_{2,i}^{t_2}\})$ with a random t_2. We can derive a new aggregated random value $e(g,g)^{t_1 t_2}$ by computing $e(g^{t_1}, g^{t_2})$ from two ciphertexts. To use the aggregated random and prevent the DDH distinguishing attack, the basic scheme should be modified. That is, a ciphertext for the first input is formed as $CT_1 = (g^{t_1} Q_1, \{(g^{x_{1,i}} W_{1,i})^{t_1} Q_{2,i}\})$ and a ciphertext for the second input is formed as $CT_2 = (g^{t_2} R_1, \{(g^{x_{2,i}} W_{2,i})^{t_2} R_{2,i}\})$ where $\mathbb{G}$ has composite order $N = p_1 p_2 p_3$, $Q \in \mathbb{G}_{p_2}$ and $R \in \mathbb{G}_{p_3}$. A private key is formed as $SK_y = \sum_{i=1}^{n} \omega_{1,i} y_{1,i} + \sum_{i=1}^{n} \omega_{2,i} y_{2,i}$. In the decryption, it should solve the discrete logarithm with a base $e(g,g)^{t_1 t_2}$.

1.3 Related Work

Single-Input Functional Encryption. The concept of functional encryption (FE) was introduced by Boneh, Sahai, and Waters [15, 16]. As mentioned before, FE allows the computation of a specific function on encrypted data by computing $f(x)$ from a ciphertext $CT(x)$ and a private key SK_f where x is a message and f is a function. The concept of FE includes identity-based encryption (IBE) [13], attribute-based encryption (ABE) [29], and predicate encryption (PE) [30]. A special FE scheme that supports an arbitrary circuit with one-private key query was proposed by Sahai and Seyalioglu [33] by using a PKE scheme and Yao's garbled circuits. Gorbunov et al. extended the one-query FE scheme of Sahai and Seyalioglu to a q-query FE scheme for all circuits by using multi-party computation (MPC) techniques [26]. After that, Goldwasser et al. showed that a single-use succinct FE scheme can be built by using a one-query FE scheme and an FHE scheme and a reusable garbling scheme can be built from this succinct FE scheme [25].

The first FE scheme for all circuits with polynomial number of private key queries was proposed by Garg et al. [21] by using indistinguishability obfuscation (iO). After this surprising result, many FE schemes with other properties were proposed by using iO [28, 35]. Recently, Garg et al. showed that a fully secure FE scheme for all circuits also can be built from multilinear maps instead of iO although a sound instantiation of multilinear maps currently does not exist [22]. If we use the power of FE that support all circuits, it is possible to achieve an adaptively secure FE scheme from a selective FE scheme and to obtain a (symmetric-key) function-private FE scheme from a non-function private FE scheme [7, 19].

Although FE schemes for all circuits can be built by using Yao's garbled circuits with universal circuits, iO, or multilinear maps, these FE schemes are relatively inefficient. Abdalla et al. showed that more practical and efficient FE schemes can be directly built from PKE schemes with special properties if the set of functions is restricted to inner product operations [2, 3]. Specifically, they showed that the ElGamal PKE scheme can be converted to an FE scheme for inner products by reusing the randomness of ciphertexts since the private keys of ElGamal are addictively homomorphic. Agrawal et al. improved these FE schemes for inner products based on ElGamal, Regev, and Paillier PKE schemes by achieving adaptive security [6]. Bishop et al. proposed a function-private FE scheme for inner products by using the dual-pairing vector space [10]. Recently, Datta et al. proposed an FE scheme for inner products with improved function privacy by using the dual system encryption technique [20].

Multi-Input Functional Encryption. Multi-input functional encryption (MI-FE) is an extension of (single-input) FE that allows the decryption algorithm takes multi-ciphertexts instead of a single-ciphertext as input. The concept of MI-FE was introduced by Goldwasser et al. [24] and they showed that MI-FE schemes can

be built from iO of Garg et al. [21] or differing-inputs obfuscation (diO). Additionally, they showed that symmetric-key MI-FE and multi-client FE (MC-FE) schemes can be constructed from iO. The application of MI-FE includes running SQL queries on encrypted database, computing on encrypted stream data, multi-client delegation of computation, and order-preserving encryption. Furthermore, it was shown that a symmetric-key MI-FE scheme can be converted into iO that is the strongest cryptographic primitive [24].

Although MI-FE is a powerful notion that has many interesting applications, many MI-FE schemes are currently built based on iO [9, 24]. Boneh et al. proposed a symmetric-key MI-FE scheme by using multilinear maps and showed that an order-revealing encryption scheme can be constructed from their MI-FE scheme [14]. An MI-FE scheme also can be built from a single-input FE scheme by converting an n-ary MI-FE scheme to an $n+1$-ary MI-FE scheme. Ananth and Jain showed that an MI-FE scheme can be built from a compact (single-key) FE scheme [8]. Brakerski et al. also showed that a symmetric-key MI-FE scheme can be built from a symmetric-key single-input FE scheme [18]. Even though an MI-FE scheme can be built from a single-input FE scheme by following the conversion methods [8, 18], the resulting MI-FE scheme is inefficient since the conversion requires an FE scheme that supports polynomial-size circuits. Concurrent to our work, Abdalla et al. also proposed a private-key MI-FE scheme for inner-product from bilinear maps [4].

Predicate Encryption for Inner Products. Predicate encryption (PE) is a specific kind of FE that supports predicates where the output of a predicate is a binary value [15]. Many efficient PE schemes were proposed in bilinear maps [1,12,17,30–32]. The PE scheme for inner products (PE-IP) of Katz, Sahai, and Waters [30] is the most expressive one among PE schemes in bilinear maps. Note that a PE-IP only outputs a binary value whether the result of inner products is zero or not, whereas an FE scheme for inner products (FE-IP) outputs the result of inner products. Shen et al. proposed a symmetric-key PE-IP scheme that provides the function privacy [34]. Agrawal et al. also proposed a PE-IP scheme based on lattices and proved its security under the LWE assumption [5]. Recently, Gorbunov et al. showed that a PE scheme for circuits can be built on lattices and proved the security under the LWE assumption [27].

2 Two-Input Functional Encryption

In this section, we define the syntax and the security model of two-input functional encryption for inner products. Next, we propose a two-input functional encryption scheme in bilinear groups and prove their security under simple assumptions.

2.1 Definition

Two-input functional encryption (TI-FE) is an extension of single-input functional encryption (FE) that allows the decryption algorithm to takes two ciphertexts as inputs instead of one ciphertext. TI-FE can be viewed as a specific kind of multi-input functional encryption (MI-FE) where the number of ciphertexts is limited to two [24]. In TI-FE for inner products, one sender creates a ciphertext CT_1 for a vector $\vec{x}_1$ and another sender creates another ciphertext CT_2 for a vector $\vec{x}_2$ independently. A receiver receives his private key $SK_{f_{\vec{y}}}$ for a function $f_{\vec{y}}$ that corresponds to a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ from a trusted center. Next, the receiver decrypts two ciphertexts CT_1 and CT_2 by using his private key $SK_{f_{\vec{y}}}$ and obtains $f_y(\vec{x}) = \langle (\vec{x}_1, \vec{x}_2), (\vec{y}_1, \vec{y}_2) \rangle$. Note that the receiver cannot obtain other information of the message $\vec{x}$ except $f_{\vec{y}}(\vec{x})$. The syntax of TI-FE is formally defined as follows:

Definition 2.1 (Two-Input Functional Encryption, TI-FE)**.** A TI-FE scheme for inner products consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

Setup$(1^\lambda, n)$. The setup algorithm takes as input a security parameter 1^λ. It outputs a master key MK, two encryption keys EK_1, EK_2, and public parameters PP.

GenKey$(\vec{y}, MK, PP)$. The key generation algorithm takes as input a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ where $\vec{y}_j = (y_{j,1}, \ldots, y_{j,n})$, the master key MK, and public parameters PP. It outputs a private key $SK_{\vec{y}}$ for the vector $\vec{y}$.

Encrypt$(j, \vec{x}_j, EK_j, PP)$. The encryption algorithm takes as input an index j, a vector $\vec{x}_j = (x_{j,1}, \ldots, x_{j,n})$, an encryption key EK_j, and public parameters PP. It outputs a ciphertext CT_j for the index j.

Decrypt$(CT_1, CT_2, SK_{\vec{y}}, PP)$. The decryption algorithm takes as input a ciphertext CT_1 for a vector $\vec{x}_1$, a ciphertext CT_2 for a vector $\vec{x}_2$, a private key $SK_{\vec{y}}$ for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$, and public parameters PP. It outputs $\langle \vec{x}, \vec{y} \rangle$ where $\vec{x} = (\vec{x}_1, \vec{x}_2)$.

The correctness property of TI-FE for inner products is defined as follows: For all MK, EK_1, EK_2, PP generated by **Setup**$(1^\lambda, n)$, any $SK_{\vec{y}}$ generated by **GenKey**$(\vec{y}, MK, PP)$ for any $\vec{y} = (\vec{y}_1, \vec{y}_2)$, and all $\vec{x}_1, \vec{x}_2$, it is required that

- **Decrypt**(**Encrypt**$(1, \vec{x}_1, EK_1, PP)$, **Encrypt**$(2, \vec{x}_2, EK_2, PP), SK_{\vec{y}}, PK) = \langle \vec{x}, \vec{y} \rangle$.

The indistinguishability-based (IND) security model of MI-FE was introduced by Goldwasser et al. [24]. By following their IND-security of MI-FE, we define the IND-security of TI-FE for inner products. In the IND-security model of TI-FE, an adversary initially submits a set of public indexes and then receives all encryption keys in the set of public indexes with public parameters. Next, the adversary can adaptively requests a private key for a vector $\vec{y}$. In the challenge step, the adversary submits two challenge vectors $\vec{x}^0 = (\vec{x}^0_1, \vec{x}^0_2)$ and $\vec{x}^1 = (\vec{x}^1_1, \vec{x}^1_2)$ with some restrictions to prevent trivial attacks and receives challenge ciphertexts CT^*_1 and CT^*_2 that are the encryption of $\vec{x}^\mu$ for a random coin $\mu \in \{0, 1\}$. If the adversary correctly guesses the coin μ, then he wins the game. In this paper, we define a weaker version of this security model. That is, the adversary should submits two challenge vectors initially before he receives the encryption keys and public parameters. The formal definition of the single-message, selective, IND-security is defined as follows:

Definition 2.2 (Single-Message, Selective, IND Security)**.** The 1-SE-IND security of a TI-FE scheme for inner products is defined in the following experiment $\mathbf{EXP}^{1\text{-}SE\text{-}IND}_{TI\text{-}FE,\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. **Init**: $\mathcal{A}$ initially submits two index sets SI and PI, two challenge vectors $\vec{x}^0 = (\vec{x}^0_1, \vec{x}^0_2)$ and $\vec{x}^1 = (\vec{x}^1_1, \vec{x}^1_2)$ where SI is a set of secret indexes, PI is a set of public indexes, and $x^c_j = (x^c_{j,1}, \ldots, x^c_{j,n})$. Note that $SI \cup PI = \{1, 2\}$ and $SI \cap PI = \emptyset$.

2. **Setup**: $\mathcal{C}$ generates a master key MK, two encryption keys EK_1, EK_2, and public parameters PP by running **Setup**$(1^\lambda, n)$. It keeps MK to itself and gives $\{EK_j\}_{j \in PI}$ and PP to $\mathcal{A}$.

3. **Query 1**: $\mathcal{A}$ adaptively requests a private key for vectors $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction that $\langle \vec{x}^1 - \vec{x}^0, \vec{y} \rangle = 0$ if $PI = \emptyset$ or $\forall j \in \{1, 2\}, \langle \vec{x}^1_j - \vec{x}^0_j, \vec{y}_j \rangle = 0$ otherwise. In response, $\mathcal{C}$ gives the private key $SK_{\vec{y}}$ to $\mathcal{A}$ by running **GenKey**$(\vec{y}, MK, PP)$.

4. **Challenge**: $\mathcal{C}$ flips a random coin $\mu \in \{0, 1\}$ and gives two challenge ciphertexts CT^*_1 and CT^*_2 to $\mathcal{A}$ by running **Encrypt**$(j, \vec{x}^\mu_j, EK_j, PP)$ for each j.

5. **Query 2**: $\mathcal{A}$ may continue to request private keys with the same restriction.

6. **Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0,1\}$ of μ. $\mathcal{C}$ outputs 1 if $\mu = \mu'$ or 0 otherwise.

A TI-FE scheme for inner products is 1-SE-IND-secure if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as $\mathbf{Adv}^{\text{1-SE-IND}}_{\text{TI-FE},\mathcal{A}}(1^\lambda) = \left|\Pr[\mathbf{EXP}^{\text{1-SE-IND}}_{\text{TI-FE},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2}\right|$ is negligible in the security parameter λ.

2.2 Bilinear Groups of Composite Order

Let $\mathcal{G}_{co}$ be a group generator algorithm that takes as input a security parameter λ and outputs a tuple $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, e)$ where p_1, p_2, p_3 are distinct primes and $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic groups of composite order $N = p_1 p_2 p_3$. Let g be a generator of $\mathbb{G}$. The bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u, v \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}_N$, $e(u^a, v^b) = e(u,v)^{ab}$.
2. Non-degeneracy: $\exists g \in \mathbb{G}$ such that $e(g,g)$ has order N in $\mathbb{G}_T$.

We say that $\mathbb{G}$ is a bilinear group if the group operations in $\mathbb{G}$ and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable in polynomial time in λ. Furthermore, we assume that the description of $\mathbb{G}$ and $\mathbb{G}_T$ includes generators of $\mathbb{G}$ and $\mathbb{G}_T$ respectively. We use the notation $\mathbb{G}_{p_i}$ to denote the subgroups of order p_i of $\mathbb{G}$ respectively. Similarly, we use the notation $\mathbb{G}_{T,p_i}$ to denote the subgroups of order p_i of $\mathbb{G}_T$ respectively.

2.3 Complexity Assumptions

We introduce simple complexity assumptions in composite-order bilinear groups for the security proofs of our TI-FE schemes.

Assumption 1 (Composite 2-Party Diffie-Hellman, C2DH)**.** Let $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, e)$ be a tuple randomly generated by $\mathcal{G}_{co}(1^\lambda)$ where $N = p_1 p_2 p_3$ is composite order of the groups. Let $g_{p_1}, g_{p_2}, g_{p_3}$ be random generators of subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. The C2DH assumption in $\mathbb{G}_{p_1 p_2}$ is that if the challenge tuple

$$D = \left((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_2}, g_{p_3}, g^a_{p_1} Q_1, g^b_{p_1} Q_2\right) \text{ and } T$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $T = T_0 = g^{ab}_{p_1} Q_3$ from $T = T_1 = g^{ab+r}_{p_1} Q_3$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{C2DH}_{\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{A}(D, T_0) = 0] - \Pr[\mathcal{A}(D, T_1) = 0]\right|$ where the probability is taken over random choices of $a, b, r \in \mathbb{Z}_N$, and $Q_1, Q_2, Q_3 \in \mathbb{G}_{p_2}$.

Remark 1. We can also define the C2DH assumption in $\mathbb{G}_{p_1 p_3}$. That is, group elements in $\mathbb{G}_{p_1}$ can be blinded by using random elements $R_i \in \mathbb{G}_{p_3}$ instead of random elements $Q_i \in \mathbb{G}_{p_2}$.

Assumption 2 (3-Party Diffie-Hellman, 3DH)**.** Let $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, e)$ be a tuple randomly generated by $\mathcal{G}_{co}(1^\lambda)$ where $N = p_1 p_2 p_3$ is composite order of the groups. Let $g_{p_1}, g_{p_2}, g_{p_3}$ be random generators of subgroups $\mathbb{G}_{p_1}, \mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ respectively. The 3DH assumption in $\mathbb{G}_{p_1}$ is that if the challenge tuple

$$D = \left((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_2}, g_{p_3}, g^a_{p_1}, g^b_{p_1}, g^c_{p_1}\right) \text{ and } T$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $T = T_0 = g^{abc}_{p_1}$ from $T = T_1 = g^{(ab+r)c}_{p_1}$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{3DH}_{\mathcal{A}}(1^\lambda) = \left|\Pr[\mathcal{A}(D, T_0) = 0] - \Pr[\mathcal{A}(D, T_1) = 0]\right|$ where the probability is taken over random choices of $a, b, c, r \in \mathbb{Z}_N$.

2.4 Construction

We propose a TI-FE scheme for inner products that is inspired by the FE scheme of Abdalla et al. [2]. Our TI-FE scheme for inner products in composite-order bilinear groups is described as follows:

TI-FE.Setup($1^\lambda, n$): This algorithm first obtains $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}_{co}(1^\lambda)$ where $N = p_1 p_2 p_3$ is composite order of the groups. Let g_1 be a random generator of the subgroup $\mathbb{G}_{p_1}$. It chooses random exponents $\{\omega_{1,i}, \omega_{2,i}\}_{i=1}^n \in \mathbb{Z}_N$ and random elements $Q, \{Q_i\}_{i=1}^n \in \mathbb{G}_{p_2}$, $R, \{R_i\}_{i=1}^n \in \mathbb{G}_{p_3}$. It outputs a master key $MK = \left(\{\omega_{1,i}, \omega_{2,i}\}_{i=1}^n\right)$, two encryption keys and public parameters as

$$EK_1 = \left(\left\{W_{1,i} = g_1^{\omega_{1,i}} Q_i\right\}_{i=1}^n\right),\ EK_2 = \left(\left\{W_{2,i} = g_1^{\omega_{2,i}} R_i\right\}_{i=1}^n\right),\ \text{and}$$
$$PP = \Big((N, \mathbb{G}, \mathbb{G}_T, e),\ g = g_1,\ Q,\ R\Big).$$

TI-FE.GenKey($\vec{y} = (\vec{y}_1, \vec{y}_2), MK, PP$): This algorithm takes as input a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ where $\vec{y}_j = (y_{j,1}, \ldots, y_{j,n})$, the master key MK, and the public parameters PP. It outputs a private key

$$SK_{\vec{y}} = \Big(\ K = \sum_{j=1}^{2} \sum_{i=1}^{n} \omega_{j,i} \cdot y_{j,i}\ \Big) \in \mathbb{Z}_N.$$

TI-FE.Encrypt($j, \vec{x}_j, EK_j, PP$): This algorithm takes as input an index $j \in \{1,2\}$, a vector $\vec{x}_j = (x_{j,1}, \ldots, x_{j,n})$, an encryption key $EK_j = (\{W_{j,i}\}_{i=1}^n)$, and the public parameters PP. It chooses a random exponent $t_j \in \mathbb{Z}_N$ and random elements $Q_1, \{Q_{2,i}\}_{i=1}^n \in \mathbb{G}_{p_2}$, and $R_1, \{R_{2,i}\}_{i=1}^n \in \mathbb{G}_{p_3}$. If $j = 1$, then it outputs a ciphertext

$$CT_1 = \Big(\ C_1 = g^{t_1} Q_1,\ \left\{D_{1,i} = \left(g^{x_{1,i}} W_{1,i}\right)^{t_1} Q_{2,i}\right\}_{i=1}^n\ \Big) \in \mathbb{G}_{p_1 p_2}^{n+1}.$$

Otherwise ($j = 2$), it outputs a ciphertext

$$CT_2 = \Big(\ E_1 = g^{t_2} R_1,\ \left\{F_{1,i} = \left(g^{x_{2,i}} W_{2,i}\right)^{t_2} R_{2,i}\right\}_{i=1}^n\ \Big) \in \mathbb{G}_{p_1 p_3}^{n+1}.$$

TI-FE.Decrypt($CT_1, CT_2, SK_{\vec{y}}, PP$): This algorithm takes as input two ciphertexts $CT_1 = (C_1, \{D_{1,i}\}_{i=1}^n)$ for a vector $\vec{x}_1 = (x_{1,1}, \ldots, x_{1,n})$ and $CT_2 = (E_1, \{F_{1,i}\}_{i=1}^n)$ for a vector $\vec{x}_2 = (x_{2,1}, \ldots, x_{2,n})$, a private key $SK_{\vec{y}} = K_1$ for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$, and the public parameters PP. It first computes the following two components

$$A = e(C_1, E_1) \quad \text{and} \quad B = \prod_{i=1}^{n} e(E_1, D_{1,i})^{y_{1,i}} \cdot \prod_{i=1}^{n} e(C_1, F_{1,i})^{y_{2,i}} \cdot A^{-K}.$$

It outputs $\langle \vec{x}, \vec{y} \rangle = \sum_{j=1}^{2} \sum_{i=1}^{n} x_{j,i} \cdot y_{j,i}$ by solving the discrete logarithm of B with a base A. Note that it only works when the inner product is relatively small.

Correctness. We show that the TI-FE scheme satisfies the correctness property defined in Definition 2.1. Since $A = e(g,g)^{t_1 t_2}$, we can derive the following equation

$$\begin{aligned} B &= \prod_{i=1}^{n} e(E_1, D_{1,i})^{y_{1,i}} \cdot \prod_{i=1}^{n} e(C_1, F_{1,i})^{y_{2,i}} \cdot A^{-K} \\ &= \prod_{i=1}^{n} \left(e(g,g)^{t_1 t_2 \cdot x_{1,i} + t_1 t_2 \cdot \omega_{1,i}}\right)^{y_{1,i}} \cdot \prod_{i=1}^{n} \left(e(g,g)^{t_1 t_2 \cdot x_{2,i} + t_1 t_2 \cdot \omega_{2,i}}\right)^{y_{2,i}} \cdot A^{-\sum_{j=1}^{2} \sum_{i=1}^{n} \omega_{j,i} y_{j,i}} \end{aligned}$$

$$= A^{\sum_{i=1}^{n} x_{1,i}y_{1,i} + \sum_{i=1}^{n} \omega_{1,i}y_{1,i}} \cdot A^{\sum_{i=1}^{n} x_{2,i}y_{2,i} + \sum_{i=1}^{n} \omega_{2,i}y_{2,i}} \cdot A^{-\sum_{j=1}^{2}\sum_{i=1}^{n} \omega_{j,i}y_{j,i}}$$
$$= A^{\sum_{j=1}^{2}\sum_{i=1}^{n} x_{j,i}y_{j,i}} = A^{\langle \vec{x}, \vec{y} \rangle}.$$

2.5 Security Analysis

To prove the security proof of our TI-FE scheme, we organize hybrid games that change the encryption of $(\vec{x}_1^0, \vec{x}_2^0)$ to the encryption of $(\vec{x}_1^1, \vec{x}_2^1)$. To simplify the proof, we first divide the type of adversaries depending on the size of PI, and then we prove the security of our TI-FE scheme depending on the type of adversaries. The detailed description of the proof is given as follows:

Theorem 2.1. *The above TI-FE scheme for inner products is 1-SE-IND-secure if the C2DH and 3DH assumptions hold.*

Proof. The security proof consists of the sequence of hybrid games. We define the games as follows:

Game $\mathbf{G}_0$. This game is the original security game except that μ is fixed to 0. That is, the challenge ciphertexts are generated for $(\vec{x}_1^0, \vec{x}_2^0)$.

Game $\mathbf{G}_F$. This final game $\mathbf{G}_F$ is almost the same with the original game except that μ is fixed to 1. That is, the challenge ciphertexts are generated for $(\vec{x}_1^1, \vec{x}_2^1)$.

Recall that SI is the set of secret indexes and PI is the set of public indexes that are submitted by an adversary. To show the indistinguishability of hybrid games $\mathbf{G}_0$ and $\mathbf{G}_F$, we divide the behavior of adversaries as three types depending on the size of PI: An adversary is Type-0 if $|PI| = 0$, Type-1 if $|PI| = 1$, and Type-2 if $|PI| = 2$.

Let $\mathbf{Adv}_{\mathcal{A}}^{G_i}$ be the advantage of $\mathcal{A}$ in a game G_i. From the Lemmas 2.2, 2.6, and 2.13, we have the following equation

$$\begin{aligned} \left|\mathbf{Adv}_{\mathcal{A}}^{G_0} - \mathbf{Adv}_{\mathcal{A}}^{G_F}\right| &\leq \left|\mathbf{Adv}_{\mathcal{A}_0}^{G_0} - \mathbf{Adv}_{\mathcal{A}_0}^{G_F}\right| + \left|\mathbf{Adv}_{\mathcal{A}_1}^{G_0} - \mathbf{Adv}_{\mathcal{A}_1}^{G_F}\right| + \left|\mathbf{Adv}_{\mathcal{A}_2}^{G_0} - \mathbf{Adv}_{\mathcal{A}_2}^{G_F}\right| \\ &\leq 2\mathbf{Adv}_{\mathcal{B}}^{3DH}(1^\lambda) + 4\mathbf{Adv}_{\mathcal{B}}^{C2DH}(1^\lambda). \end{aligned}$$

This completes our proof. □

2.5.1 Type-0 Adversary

Lemma 2.2. *If the 3DH assumption holds, then no polynomial-time Type-0 adversary can distinguish between $\mathbf{G}_0$ and $\mathbf{G}_F$ with a non-negligible advantage.*

Proof. If an adversary is Type-0, then he can request any private key for $\vec{y} = (\vec{y}_1, \vec{y}_2)$ such that $\sum_{j=1}^{2}\langle \vec{x}_j^1 - \vec{x}_j^0, \vec{y}_j \rangle = 0$ since the scheme is defined in the private-key setting where all encryption keys are hidden. In case of Type-0, we should be careful that the adversary can query a private key for $\vec{y} = (\vec{y}_1, \vec{y}_2)$ that satisfies $\langle \vec{x}_j^1 - \vec{x}_j^0, \vec{y}_j \rangle \neq 0$ for some $j \in \{1,2\}$. Because of these queries, the security proof for the Type-0 adversary is relatively complicated, but we solve this problem by organizing hybrid games that change the challenge ciphertexts for two indexes at the same time and by introducing a new complexity assumption. For the Type-0 adversary, we define hybrid games $\mathbf{H}_0 = \mathbf{G}_0, \mathbf{H}_1, \mathbf{H}_1'$, and $\mathbf{H}_2 = \mathbf{G}_F$. The games are defined as follows:

Game $\mathbf{H}_0$. This game is the same as $\mathbf{G}_0$. That is, the challenge ciphertexts are generated for $(\vec{x}_1^0, \vec{x}_2^0)$.

Game $\mathbf{H}_1$. In this game, the challenge ciphertexts are generated for $(\vec{x}_1^0 + r \cdot (\vec{x}_1^1 - \vec{x}_1^0),\ \vec{x}_2^0 + r \cdot (\vec{x}_2^1 - \vec{x}_2^0))$ where $r \in \mathbb{Z}_N$ is a hidden random value.

Game $\mathbf{H}_1'$. This game is almost similar to $\mathbf{H}_1$ except that the challenge ciphertexts are generated for $(\vec{x}_1^1 + r' \cdot (\vec{x}_1^1 - \vec{x}_1^0),\ \vec{x}_2^1 + r' \cdot (\vec{x}_2^1 - \vec{x}_2^0))$ where $r' \in \mathbb{Z}_N$ is a hidden random value.

Game $\mathbf{H}_2$. This game is the same as $\mathbf{G}_F$. That is, the challenge ciphertexts are generated for $(\vec{x}_1^1, \vec{x}_2^1)$.

Let $\mathbf{Adv}_{\mathcal{A}}^{H_i}$ be the advantage of $\mathcal{A}_0$ in a game H_i. From the Lemmas 2.3, 2.4, and 2.5, we have the following equation

$$\begin{aligned}\left|\mathbf{Adv}_{\mathcal{A}_0}^{G_0} - \mathbf{Adv}_{\mathcal{A}_0}^{G_F}\right| &\leq \left|\mathbf{Adv}_{\mathcal{A}_0}^{H_0} - \mathbf{Adv}_{\mathcal{A}_0}^{H_1}\right| + \left|\mathbf{Adv}_{\mathcal{A}_0}^{H_1} - \mathbf{Adv}_{\mathcal{A}_0}^{H_1'}\right| + \left|\mathbf{Adv}_{\mathcal{A}_0}^{H_1'} - \mathbf{Adv}_{\mathcal{A}_0}^{H_2}\right| \\ &\leq 2\mathbf{Adv}_{\mathcal{B}}^{3DH}(1^\lambda).\end{aligned}$$

This completes our proof. □

Lemma 2.3. *If the 3DH assumption holds, then no polynomial-time Type-0 adversary can distinguish $\boldsymbol{H}_0$ from $\boldsymbol{H}_1$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}_0$ that breaks the security game with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the 3DH assumption in the subgroup $\mathbb{G}_{p_1}$ using $\mathcal{A}_0$ is given: a challenge tuple $D = \left((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_2}, g_{p_3}, g_{p_1}^a, g_{p_1}^b, g_{p_1}^c\right)$ and T where $T = g_{p_1}^{abc}$ or $T = g_{p_1}^{(ab+r)c}$. Then $\mathcal{B}$ interacts with $\mathcal{A}_0$ as follows:

Init: $\mathcal{A}$ submits two sets $SI = \{1,2\}$ and $PI = \emptyset$, and two challenge vectors $\vec{x}^0 = (\vec{x}_1^0, \vec{x}_2^0)$ and $\vec{x}^1 = (\vec{x}_1^1, \vec{x}_2^1)$ where $\vec{x}_j^c = (x_{j,1}^c, \ldots, x_{j,n}^c)$.

Setup: $\mathcal{B}$ first selects random $\{w_{1,i}', w_{2,i}'\}_{i=1}^n \in \mathbb{Z}_N$ and implicitly sets the master key $\left\{\omega_{1,i} = ab(x_{1,i}^1 - x_{1,i}^0) + w_{1,i}', \omega_{2,i} = ab(x_{2,i}^1 - x_{2,i}^0) + w_{2,i}'\right\}_{i=1}^n$. It creates public parameters $PP = \left((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Q = g_{p_2}, R = g_{p_3}\right)$. Note that it cannot create two encryption keys since $g_{p_1}^{ab}$ is not given.

Query 1: $\mathcal{A}_0$ may adaptively request a private key for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction $\langle \vec{x}^1 - \vec{x}^0, \vec{y} \rangle = 0$. $\mathcal{B}$ creates the private key $SK_{\vec{y}} = \left(K = \sum_{i=1}^n w_{1,i}' y_{1,i} + \sum_{i=1}^n w_{2,i}' y_{2,i}\right)$. Note that it can create this private key because of the restriction.

Challenge: In the challenge step, $\mathcal{B}$ first selects random $d \in \mathbb{Z}_N, Q_1, \{Q_{2,i}\} \in \mathbb{G}_{p_2}$, $R_1, \{R_{2,i}\} \in \mathbb{G}_{p_3}$. Next, it implicitly sets $t_1 = c, t_2 = cd$ and creates challenge ciphertexts

$$\begin{aligned}CT_1^* &= \left(C_1 = g_{p_1}^c Q_1,\ \left\{D_{1,i} = \left(g_{p_1}^c\right)^{x_{1,i}^0} (T)^{(x_{1,i}^1 - x_{1,i}^0)} \left(g_{p_1}^c\right)^{w_{1,i}'} Q_{2,i}\right\}_{i=1}^n\right), \\ CT_2^* &= \left(E_1 = (g_{p_1}^c)^d R_1,\ \left\{F_{1,i} = \left((g_{p_1}^c)^d\right)^{x_{2,i}^0} (T^d)^{(x_{2,i}^1 - x_{2,i}^0)} \left((g_{p_1}^c)^d\right)^{w_{2,i}'} R_{2,i}\right\}_{i=1}^n\right).\end{aligned}$$

If $T = T_0 = g_{p_1}^{abc}$, then the challenge ciphertexts are well formed as the same as the game $\mathbf{H}_0$. If $T = T_1 = g_{p_1}^{(ab+r)c}$, then the challenge ciphertexts are the encryption of $\vec{x}_1^0 + r(\vec{x}_1^1 - \vec{x}_1^0)$ and $\vec{x}_2^0 + r(\vec{x}_2^1 - \vec{x}_2^0)$ as the same as the game $\mathbf{H}_1$.

Query 2: Same as Query 1.

Guess: Finally, $\mathcal{A}_0$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

Lemma 2.4. *No Type-0 adversary can distinguish $\boldsymbol{H}_1$ from $\boldsymbol{H}_1'$.*

Proof. To prove this lemma, we show that the challenge ciphertexts CT_1^* and CT_2^* that are the encryption of $\vec{x}_1^0 + r \cdot (\vec{x}_1^1 - \vec{x}_1^0)$ and $\vec{x}_2^0 + r \cdot (\vec{x}_2^1 - \vec{x}_2^0)$ can be restated as the encryption of $\vec{x}_1^1 + r' \cdot (\vec{x}_1^1 - \vec{x}_1^0)$ and $\vec{x}_2^1 + r' \cdot (\vec{x}_2^1 - \vec{x}_2^0)$ where r and r' are hidden to the adversary. By simply setting $r = r' + 1$, we obtain following equations

$$\vec{x}_1^0 + r \cdot (\vec{x}_1^1 - \vec{x}_1^0) = \vec{x}_1^0 + (r'+1) \cdot (\vec{x}_1^1 - \vec{x}_1^0) = \vec{x}_1^1 + r' \cdot (\vec{x}_1^1 - \vec{x}_1^0),$$
$$\vec{x}_2^0 + r \cdot (\vec{x}_2^1 - \vec{x}_2^0) = \vec{x}_2^0 + (r'+1) \cdot (\vec{x}_2^1 - \vec{x}_2^0) = \vec{x}_2^1 + r' \cdot (\vec{x}_2^1 - \vec{x}_2^0).$$

Note that the private key $SK_{\vec{y}}$ cannot be used to distinguish the change since $\langle \vec{x}^1 - \vec{x}^0, \vec{y} \rangle = 0$ by the restriction of the security model. □

Lemma 2.5. *If the 3DH assumption holds, then no polynomial-time Type-0 adversary can distinguish $\boldsymbol{H}_1'$ from $\boldsymbol{H}_2$ with a non-negligible advantage.*

Proof. The proof of this lemma is symmetric to that of Lemma 2.3 except that the challenge ciphertexts are the encryption of $(\vec{x}_1^1, \vec{x}_2^1)$ instead of $(\vec{x}_1^0, \vec{x}_2^0)$. □

2.5.2 Type-1 Adversary

Lemma 2.6. *If the C2DH assumption in $\mathbb{G}_{p_1 p_2}$ and the C2DH assumption in $\mathbb{G}_{p_1 p_3}$ hold, then no polynomial-time Type-1 adversary can distinguish between $\boldsymbol{G}_0$ and $\boldsymbol{G}_F$ with a non-negligible advantage.*

Proof. If an adversary is Type-1, then one encryption key is hidden and another encryption key is published. Thus, the adversary only can query a private key for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction $\langle \vec{x}_j^1 - \vec{x}_j^0, \vec{y}_j \rangle = 0$ for all $j \in \{1,2\}$ since he can create a ciphertext for the public index by using the published encryption key. Because of this strong restriction, we can organize additional hybrid games that change challenge ciphertexts one by one. For the Type-1 adversary, we define hybrid games $\mathbf{I}_0 = \mathbf{G}_0, \mathbf{I}_1, \mathbf{I}_1', \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}_3'$, and $\mathbf{I}_4 = \mathbf{G}_F$. The games are defined as follows:

Game $\mathbf{I}_0$. This game is the same as $\mathbf{G}_0$. That is, the challenge ciphertexts are generated for $(\vec{x}_1^0, \vec{x}_2^0)$.

Game $\mathbf{I}_1$. In this game, the challenge ciphertexts are generated for $(\vec{x}_1^0 + r_1 \cdot (\vec{x}_1^1 - \vec{x}_1^0),\ \vec{x}_2^0)$ where $r_1 \in \mathbb{Z}_N$ is a hidden random value.

Game $\mathbf{I}_1'$. This game is almost similar to $\mathbf{I}_1$ except that the challenge ciphertexts are generated for $(\vec{x}_1^1 + r_1' \cdot (\vec{x}_1^1 - \vec{x}_1^0),\ \vec{x}_2^0)$ where $r_1' \in \mathbb{Z}_N$ is a hidden random value.

Game $\mathbf{I}_2$. In this game $\mathbf{I}_2$, the challenge ciphertexts are generated for $(\vec{x}_1^1,\ \vec{x}_2^0)$.

Game $\mathbf{I}_3$. In this game, the challenge ciphertexts are generated for $(\vec{x}_1^1,\ \vec{x}_2^0 + r_2 \cdot (\vec{x}_2^1 - \vec{x}_2^0))$ where $r_2 \in \mathbb{Z}_N$ is a hidden random value.

Game $\mathbf{I}_3'$. This game is almost similar to $\mathbf{I}_3$ except that the challenge ciphertexts are generated for $(\vec{x}_1^1,\ \vec{x}_2^1 + r_2' \cdot (\vec{x}_2^1 - \vec{x}_2^0))$ where $r_2' \in \mathbb{Z}_N$ is a hidden random value.

Game $\mathbf{I}_4$. This game is the same as $\mathbf{G}_4$. That is, the challenge ciphertexts are generated for $(\vec{x}_1^1, \vec{x}_2^1)$.

Let $\mathbf{Adv}_{\mathcal{A}_1}^{I_i}$ be the advantage of $\mathcal{A}_1$ in a game I_i. From the Lemmas 2.7, 2.8, 2.9, 2.10, 2.11, and 2.12, we have the following equation

$$\begin{aligned}\left|\mathbf{Adv}_{\mathcal{A}_1}^{G_0} - \mathbf{Adv}_{\mathcal{A}_1}^{G_F}\right| &\leq \left|\mathbf{Adv}_{\mathcal{A}_1}^{I_0} - \mathbf{Adv}_{\mathcal{A}_1}^{I_1}\right| + \left|\mathbf{Adv}_{\mathcal{A}_1}^{I_1} - \mathbf{Adv}_{\mathcal{A}_1}^{I_1'}\right| + \left|\mathbf{Adv}_{\mathcal{A}_1}^{I_1'} - \mathbf{Adv}_{\mathcal{A}_1}^{I_2}\right| + \\ &\quad \left|\mathbf{Adv}_{\mathcal{A}_1}^{I_2} - \mathbf{Adv}_{\mathcal{A}_1}^{I_3}\right| + \left|\mathbf{Adv}_{\mathcal{A}_1}^{I_3} - \mathbf{Adv}_{\mathcal{A}_1}^{I_3'}\right| + \left|\mathbf{Adv}_{\mathcal{A}_1}^{I_3'} - \mathbf{Adv}_{\mathcal{A}_1}^{I_4}\right| \\ &\leq 4\mathbf{Adv}_{\mathcal{B}}^{C2DH}(1^\lambda).\end{aligned}$$

This completes our proof. □

Lemma 2.7. *If the C2DH assumption in $\mathbb{G}_{p_1p_2}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}_0$ from $\boldsymbol{I}_1$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}_1$ that breaks the security game with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the C2DH assumption in $\mathbb{G}_{p_1p_2}$ using $\mathcal{A}_1$ is given: a challenge tuple $D = \big((N,\mathbb{G},\mathbb{G}_T,e), g_{p_1}, g_{p_2}, g_{p_3}, g^a_{p_1}Q_1, g^b_{p_1}Q_2\big)$ and T where $T = g^{ab}_{p_1}Q_3$ or $T = g^{(a+r)b}_{p_1}Q_3$. Then $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Init: $\mathcal{A}_1$ submits two sets SI and PI, and two challenge vectors $\vec{x}^0 = (\vec{x}^0_1, \vec{x}^0_2)$ and $\vec{x}^1 = (\vec{x}^1_1, \vec{x}^1_2)$ where $|SI| = |PI| = 1$ and $\vec{x}^c_j = (x^c_{j,1}, \ldots, x^c_{j,n})$.

Setup: $\mathcal{B}$ first selects random $\{w'_{1,i}, \omega_{2,i}\}^n_{i=1} \in \mathbb{Z}_N$, $\{Q_i\}^n_{i=1} \in \mathbb{G}_{p_2}, \{R_i\}^n_{i=1} \in \mathbb{G}_{p_3}$ and creates encryption keys $EK_1 = \big(\{W_{1,i} = (g^a_{p_1}Q_1)^{(x^1_{1,i}-x^0_{1,i})} g^{w'_{1,i}}_{p_1} Q_i\}^n_{i=1}\big)$ and $EK_2 = \big(\{W_{2,i} = g^{\omega_{2,i}}_{p_1} R_i\}^n_{i=1}\big)$ by implicitly setting $\big\{\omega_{1,i} = a(x^1_{1,i} - x^0_{1,i}) + w'_{1,i}\big\}^n_{i=1}$. Next, it creates $PP = \big((N,\mathbb{G},\mathbb{G}_T,e), g = g_{p_1}, Q = g_{p_2}, R = g_{p_3}\big)$ and gives $EK_{j\in PI}, PP$ to $\mathcal{A}_1$. Note that $\mathcal{B}$ can create EK_2 but it is not helpful because of the strong restriction.

Query 1: $\mathcal{A}_1$ may adaptively request a private key for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction $\langle \vec{x}^1_j - \vec{x}^0_j, \vec{y}_j \rangle = 0$ for $j \in \{1,2\}$. $\mathcal{B}$ creates the private key $SK_{\vec{y}} = \big(K = \sum^n_{i=1} w'_{1,i} y_{1,i} + \sum^n_{i=1} \omega_{2,i} y_{2,i}\big)$.

Challenge: $\mathcal{B}$ first creates CT^*_2 by running $\mathbf{Encrypt}(2, \vec{x}^0_2, EK_2, PP)$ since it knows EK_2. Next, it selects random $Q'_1, \{Q'_{2,i}\}^n_{i=1} \in \mathbb{G}_{p_2}$ and creates the challenge ciphertext by implicitly setting $t_1 = b$ as

$$CT^*_1 = \Big(C_1 = \big(g^b_{p_1}Q_2\big)Q'_1,\ \big\{D_{1,i} = \big(g^b_{p_1}Q_2\big)^{x^0_{1,i}} (T)^{(x^1_{1,i}-x^0_{1,i})} \big(g^b_{p_1}Q_2\big)^{w'_{1,i}} Q'_{2,i},\ \big\}^n_{i=1}\Big).$$

Query 2: Same as Query 1.

Guess: Finally, $\mathcal{A}_1$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

Lemma 2.8. *No Type-1 adversary can distinguish $\boldsymbol{I}_1$ from $\boldsymbol{I}'_1$.*

Proof. The proof of this lemma is similar to that of Lemma 2.4. We show that the challenge ciphertext CT^*_1 that is the encryption of $\vec{x}^0_1 + r \cdot (\vec{x}^1_1 - \vec{x}^0_1)$ can be restated as the encryption of $\vec{x}^1_1 + r' \cdot (\vec{x}^1_1 - \vec{x}^0_1)$ where r and r' are hidden to the adversary. By simply setting $r = r' + 1$, we obtain following equations

$$\vec{x}^0_1 + r \cdot (\vec{x}^1_1 - \vec{x}^0_1) = \vec{x}^0_1 + (r'+1) \cdot (\vec{x}^1_1 - \vec{x}^0_1) = \vec{x}^1_1 + r' \cdot (\vec{x}^1_1 - \vec{x}^0_1).$$

Note that the private key $SK_{\vec{y}}$ cannot be used to distinguish the change since $\langle \vec{x}^1_j - \vec{x}^0_j, \vec{y}_j \rangle = 0$ for all $j \in \{1,2\}$ by the strong restriction. □

Lemma 2.9. *If the C2DH assumption in $\mathbb{G}_{p_1p_2}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}'_1$ from $\boldsymbol{I}_2$ with a non-negligible advantage.*

Proof. The proof of this lemma is symmetric to that of Lemma 2.7. □

Lemma 2.10. *If the C2DH assumption in $\mathbb{G}_{p_1p_3}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}_2$ from $\boldsymbol{I}_3$ with a non-negligible advantage.*

Proof. The proof of this lemma is very similar to that of Lemma 2.7 except that the C2DH assumption in $\mathbb{G}_{p_1p_3}$ is used to embed itself in the index 2.

Suppose there exists an adversary $\mathcal{A}_1$ that breaks the security game with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the C2DH assumption in $\mathbb{G}_{p_1p_3}$ using $\mathcal{A}$ is given: a challenge tuple $D = \big((N,\mathbb{G},\mathbb{G}_T,e), g_{p_1},g_{p_2},g_{p_3},g^a_{p_1}R_1,g^b_{p_1}R_2\big)$ and T where $T = g^{ab}_{p_1}R_3$ or $T = g^{(a+r)b}_{p_1}R_3$. Then $\mathcal{B}$ interacts with $\mathcal{A}_1$ as follows:

Init: $\mathcal{A}$ submits two sets SI and PI, and two challenge vectors $\vec{x}^0 = (\vec{x}^0_1,\vec{x}^0_2)$ and $\vec{x}^1 = (\vec{x}^1_1,\vec{x}^1_2)$ where $SI = \{2\}$, $PI = \{1\}$, and $\vec{x}^c_j = (x^c_{j,1},\ldots,x^c_{j,n})$.

Setup: $\mathcal{B}$ first selects random $\{\omega_{1,i},w'_{2,i}\}^n_{i=1} \in \mathbb{Z}_N$, $\{Q_i\}^n_{i=1} \in \mathbb{G}_{p_2}, \{R_i\}^n_{i=1} \in \mathbb{G}_{p_3}$ and creates encryption keys $EK_1 = \big(\{W_{1,i} = g^{\omega_{1,i}}_{p_1}Q_i\}^n_{i=1}\big)$ and $EK_2 = \big(\{W_{2,i} = (g^a_{p_1}R_1)^{(x^1_{2,i}-x^0_{2,i})}g^{w'_{2,i}}_{p_1}R_i\}^n_{i=1}\}^n_{i=1}\big)$ by implicitly setting $\big\{\omega_{2,i} = a(x^1_{2,i}-x^0_{2,i})+w'_{2,i}\big\}^n_{i=1}$. Next, it creates $PP = \big((N,\mathbb{G},\mathbb{G}_T,e),g = g_{p_1},Q = g_{p_2},R = g_{p_3}\big)$ and gives $EK_{j\in PI},PP$ to $\mathcal{A}_1$. Note that $\mathcal{B}$ can create EK_1 but it is not helpful because of the strong restriction.

Query 1: $\mathcal{A}$ may adaptively request a private key for a vector $\vec{y} = (\vec{y}_1,\vec{y}_2)$ with the restriction $\langle\vec{x}^1_j - \vec{x}^0_j,\vec{y}_j\rangle = 0$ for $j \in \{1,2\}$. $\mathcal{B}$ creates the private key $SK_{\vec{y}} = \big(K = \sum^n_{i=1}\omega_{1,i}y_{1,i} + \sum^n_{i=1}w'_{2,i}y_{2,i}\big)$.

Challenge: $\mathcal{B}$ first creates CT^*_1 by running $\mathbf{Encrypt}(1,\vec{x}^1_1,EK_1,PP)$ since it knows EK_1. Next, it selects random $R_1,\{R_{2,i}\}^n_{i=1} \in \mathbb{G}_{p_3}$ and creates the challenge ciphertext by implicitly setting $t_2 = b$ as

$$CT^*_2 = \Big(E_1 = \big(g^b_{p_1}R_2\big)R'_1,\ \big\{F_{1,i} = \big(g^b_{p_1}R_2\big)^{x^0_{2,i}}(T)^{(x^1_{2,i}-x^0_{2,i})}\big(g^b_{p_1}R_2\big)^{w'_{2,i}}R'_{2,i}\big\}^n_{i=1}\Big).$$

Query 2: Same as Query 1.

Guess: Finally, $\mathcal{A}_1$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

Lemma 2.11. *No Type-1 adversary can distinguish $\boldsymbol{I}_3$ from $\boldsymbol{I}'_3$.*

Proof. The proof of this lemma is almost the same as that of Lemma 2.8 except that CT^*_2 is considered instead of CT^*_1. □

Lemma 2.12. *If the C2DH assumption in $\mathbb{G}_{p_1p_3}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}'_3$ from $\boldsymbol{I}_4$ with a non-negligible advantage.*

Proof. The proof of this lemma is symmetric to that of Lemma 2.10. □

2.5.3 Type-2 Adversary

Lemma 2.13. *If the C2DH assumption in $\mathbb{G}_{p_1p_2}$ and the C2DH assumption in $\mathbb{G}_{p_1p_3}$ hold, then no polynomial-time Type-2 adversary can distinguish between $\boldsymbol{G}_0$ and $\boldsymbol{G}_F$ with a non-negligible advantage.*

Proof. If an adversary if Type-2, then the scheme is defined in the public-key setting where all encryption keys are published. Because of these published encryption keys, the adversary can create any ciphertext CT_j for any vector $\vec{x}_j$. Therefore, the adversary can query any private key for a vector $\vec{y} = (\vec{y}_1,\vec{y}_2)$ with the restriction $\langle\vec{x}_j,\vec{y}_j\rangle = 0$ for all $j \in \{1,2\}$. Recall that this restriction is the same as that of the Type-1 adversary. Because of this strong restriction, we can also organize hybrid games that change challenge ciphertexts one by one. For the Type-2 adversary, we also use the same hybrid games $\mathbf{I}_0,\mathbf{I}_1,\mathbf{I}'_1,\mathbf{I}_2,\mathbf{I}_3,\mathbf{I}'_3$, and $\mathbf{I}_4$ that are defined for the Type-1 adversary.

The proofs of all lemmas are almost the same as those of lemmas in Lemma 2.6 for the Type-1 adversary except that all encryption keys should be given to the adversary. Note that the simulator for the Type-1 adversary can create all encryption keys EK_1,EK_2 since these keys are not helpful by the strong restriction in the security model. We omit the details of all lemmas for the Type-2 adversary. □

2.6 Discussions

TI-FE in the Public-Key Setting. We can obtain a TI-FE scheme for inner products in the public-key setting if all encryption keys of our TI-FE scheme are revealed to an adversary. In the public-key setting, the adversary can obtain additional information about the message since he can create any ciphertext for any index by using an encryption key for the index. As pointed by Goldwasser et al. [24], an MI-FE scheme in the public-key setting supports only a limited set of functions because of the revealed encryption keys. Because of this limitation in the public-key settings, an MI-FE scheme for inner products can be easily built from a single-input FE scheme for inner products by simply using multiple instances of single-input FE for inner products. Note that an adversary always can learn $\langle \vec{x}_j, \vec{y}_j \rangle$ for any index j in MI-FE for inner products. Thus, an MI-FE scheme for inner products is straightforward from an FE for inner products.

TI-FE in the Secret-Key Setting. If all encryption keys of our TI-FE scheme are not revealed to an adversary, then a TI-FE scheme in the secret-key setting is obtained. In case of the secret-key setting, we can convert our TI-FE scheme in composite-order bilinear groups into a TI-FE scheme in prime-order (symmetric) bilinear groups. The main reason of this conversion is that it is relatively hard for the adversary to obtain some information of a message by using pairing operations if all encryption keys are hidden to the adversary. Recall that additional random subgroup elements in composite-order bilinear groups were used to randomize the components of ciphertexts in our TI-FE scheme.

3 Two-Client Functional Encryption

In this section, we first define the syntax and the security model of two-client functional encryption. Next, we propose a two-client functional encryption scheme in composite-order bilinear groups and prove its selective security under simple assumptions.

3.1 Definition

The concept of multi-client functional encryption (MC-FE) was introduced by Goldwasser et al. [24]. We define two-client functional encryption (TC-FE) for inner products by following their definition. A TC-FE scheme is very similar to a TI-FE scheme except that each client who has an individual encryption key can create a ciphertext associated with a time period. That is, one client can create a ciphertext $CT_{1,T}$ for a message x_1 with a time period T by using his encryption key EK_1 and another client also can create a ciphertext $CT_{2,T'}$ for a message x_2 with a time period T' by using his encryption key EK_2. If the time periods T, T' of two ciphertexts are equal, then a receiver who has a private key SK_f can computes $f(x_1, x_2)$ from two ciphertexts. The formal syntax of TC-FE is defined as follows:

Definition 3.1 (Two-Client Functional Encryption, TC-FE)**.** A TC-FE scheme for inner products consists of four algorithms **Setup**, **GenKey**, **Encrypt**, and **Decrypt**, which are defined as follows:

Setup$(1^\lambda, n)$. The setup algorithm takes as input a security parameter 1^λ. It outputs a master key MK, two encryption keys EK_1, EK_2, and public parameters PP.

GenKey$(\vec{y}, MK, PP)$. The key generation algorithm takes as input a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ where $\vec{y}_j = (y_{j,1}, \ldots, y_{j,n})$, the master key MK, and public parameters PP. It outputs a private key $SK_{\vec{y}}$ for the vector $\vec{y}$.

Encrypt$(j, \vec{x}_j, T, EK_j, PP)$. The encryption algorithm takes as input an index $j \in \{1,2\}$, a vector $\vec{x}_j = (x_{j,1}, \ldots, x_{j,n})$, time T, an encryption key EK_j, and the master key MK. It outputs a ciphertext $CT_{j,T}$ for the index j and time T.

Decrypt$(CT_{1,T}, CT_{2,T'}, SK_{\vec{y}}, PP)$. The decryption algorithm takes as input a ciphertext $CT_{1,T}$ for a vector $\vec{x}_1$ and time T, a ciphertext $CT_{2,T'}$ for a vector $\vec{x}_2$ and time T', a private key $SK_{\vec{y}}$ for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$, and public parameters PP. It outputs $\langle \vec{x}, \vec{y} \rangle$ where $\vec{x} = (\vec{x}_1, \vec{x}_2)$.

The correctness property of TC-FE for inner products is defined as follows: For all $MK, \{EK_j\}, PP$ generated by **Setup**$(1^\lambda, n)$, any $SK_{\vec{y}}$ generated by **GenKey**$(\vec{y}, MK, PP)$ for any $\vec{y} = (\vec{y}_1, \vec{y}_2)$, all $\vec{x}_1, \vec{x}_2$, and any T, T', it is required that

- If $T = T'$, then **Decrypt**(**Encrypt**$(1, \vec{x}_1, T, EK_1, PP)$, **Encrypt**$(2, \vec{x}_2, T', EK_j, PP), SK_{\vec{y}}, PK) = \langle \vec{x}, \vec{y} \rangle$.
- If $T \neq T'$, then it outputs $\perp$.

The IND-security of MC-FE was also defined by Goldwasser et al. [24]. We define the IND-security of TC-FE by following their definition of the IND-security. For the security proof of our TC-FE scheme, we define the single-message, selective, IND-security (1-SE-IND) of TC-FE. In the 1-SE-IND security model, an adversary first submits a set of public indexes, two challenge vectors $\vec{x}^0, \vec{x}^1$, and a challenge time period T^*, and then he can receives all encryption keys in the set of public indexes and public parameters. After that the adversary requests a private key for a vector with some restrictions. Additionally, the adversary can query a ciphertext for a time period $T \neq T^*$ with some restrictions. In the challenge step, the adversary receives challenge ciphertexts that are encryption of $\vec{x}^\mu$ for a random μ and wins the game if he correctly guesses the challenge. The formal definition of the 1-SE-IND security is defined as follows:

Definition 3.2 (Single-Message, Selective, IND Security)**.** The 1-SE-IND security of a TC-FE scheme for inner products is defined in the following experiment $\mathbf{EXP}^{\text{1-SE-IND}}_{\text{TC-FE},\mathcal{A}}(1^\lambda)$ between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

1. **Init**: $\mathcal{A}$ first submits two index sets SI and PI, two challenge vectors $\vec{x}^0 = (\vec{x}^0_1, \vec{x}^0_2)$ and $\vec{x}^1 = (\vec{x}^1_1, \vec{x}^1_2)$, and a challenge time period T^* where SI is a set of secret indexes, PI is a set of public indexes, and $x^c_j = (x^c_{j,1}, \ldots, x^c_{j,n})$. Note that $SI \cup PI = \{1, 2\}$ and $SI \cap PI = \emptyset$.
2. **Setup**: $\mathcal{C}$ generates a master key MK, two encryption keys EK_1, EK_2, and public parameters PP by running **Setup**$(1^\lambda, n)$. It keeps MK to itself and gives $\{EK_j\}_{j \in PI}$, PP to $\mathcal{A}$.
3. **Query 1**: $\mathcal{A}$ adaptively requests private key and ciphertext queries, where each query is one of two types:
   - If this is a private key query for vectors $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction that $\langle \vec{x}^1 - \vec{x}^0, \vec{y} \rangle = 0$ if $PI = \emptyset$ or $\forall j \in \{1, 2\}, \langle \vec{x}^1_j - \vec{x}^0_j, \vec{y}_j \rangle = 0$ otherwise, then $\mathcal{C}$ gives the private key $SK_{\vec{y}}$ to $\mathcal{A}$ by running **GenKey**$(\vec{y}, MK, PP)$.
   - If this is a ciphertext query for an index j, a vector $\vec{x}_j$, and a time period T with the restriction that $j \in SI$ and $T \neq T^*$, then $\mathcal{C}$ gives the ciphertext $CT_{j,T}$ to $\mathcal{A}$ by running **Encrypt**$(j, \vec{x}_j, T, EK_j, PP)$.
4. **Challenge**: $\mathcal{C}$ flips a random coin $\mu \in \{0, 1\}$ and gives two challenge ciphertexts CT^*_{1,T^*} and CT^*_{2,T^*} to $\mathcal{A}$ by running **Encrypt**$(j, \vec{x}^\mu_j, T^*, EK_j, PP)$ for each j.
5. **Query 2**: $\mathcal{A}$ may continue to request private keys and ciphertexts with the same restriction.
6. **Guess**: $\mathcal{A}$ outputs a guess $\mu' \in \{0, 1\}$ of μ. $\mathcal{C}$ outputs 1 if $\mu = \mu'$ or 0 otherwise.

A TC-FE scheme for inner products is 1-SE-IND-secure if for all PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ defined as $\mathbf{Adv}^{\text{1-SE-IND}}_{\text{TC-FE},\mathcal{A}}(1^\lambda) = \left| \Pr[\mathbf{EXP}^{\text{1-SE-IND}}_{\text{TC-FE},\mathcal{A}}(1^\lambda) = 1] - \frac{1}{2} \right|$ is negligible in the security parameter λ.

3.2 Construction

The design idea of a TC-FE scheme is to start from our TI-FE scheme and add additional structures to handle a time period in ciphertexts. To handle a time period, we add the ciphertext structure $CT_{IBE} = (g^t, (u^T h)^t)$ and the private key structure $SK_{IBE} = (g^{\alpha}(u^T h)^r, g^{-r})$ of the Boneh-Boyen IBE (BB-IBE) scheme [11] to our TI-FE scheme. In this case, if the time periods of two ciphertexts are equal, then the normal decryption process of the TI-FE scheme can be performed after removing additional components for time periods by running the IBE decryption. Additionally, we can use the proof technique of the BB-IBE scheme to create a ciphertext for a time period $T \neq T^*$ where T^* is the challenge time period. The detailed description of our TC-FE scheme is described as follows:

TC-FE.Setup$(1^{\lambda}, n)$**:** This algorithm first obtains $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, e)$ by running $\mathcal{G}_{co}(1^{\lambda})$ where $N = p_1 p_2 p_3$ is composite order of the groups. Let g_1 be a random generator of the subgroup $\mathbb{G}_{p_1}$. It chooses random exponents $\{\omega_{1,i}, \omega_{2,i}\}_{i=1}^n \in \mathbb{Z}_N$. It also selects random elements $u_1, h_1, u_2, h_2 \in \mathbb{G}_{p_1}$, $Q, \{Q_i\}_{i=1}^n \in \mathbb{G}_{p_2}$, and $R, \{R_i\}_{i=1}^n \in \mathbb{G}_{p_3}$. It outputs a master key $MK = \left(\{\omega_{1,i}, \omega_{2,i}\}_{i=1}^n\right)$ and two encryption keys and public parameters as

$$EK_1 = \left(\{W_{1,i} = g_1^{\omega_{1,i}} Q_i\}_{i=1}^n \right),\ EK_2 = \left(\{W_{2,i} = g_1^{\omega_{2,i}} R_i\}_{i=1}^n \right),$$
$$PP = \left((N, \mathbb{G}, \mathbb{G}_T, e),\ g = g_1,\ Q,\ R,\ u_1,\ h_1,\ u_2,\ h_2 \right).$$

TC-FE.GenKey$(\vec{y} = (\vec{y}_1, \vec{y}_2), MK, PP)$**:** This algorithm takes as input a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ where $\vec{y}_j = (y_{j,1}, \ldots, y_{j,n})$, the master key MK, and the public parameters PP. It outputs a private key

$$SK_{\vec{y}} = \left(K = \sum_{j=1}^{2} \sum_{i=1}^{n} \omega_{j,i} \cdot y_{j,i} \right) \in \mathbb{Z}_N.$$

TC-FE.Encrypt$(j, \vec{x}_j, T, EK_j, PP)$**:** This algorithm takes as input an index $j \in \{1,2\}$, a vector $\vec{x}_j = (x_{j,1}, \ldots, x_{j,n})$, a time period T, an encryption key EK_j, and the public parameters PP. It chooses random exponents $t_j, \{r_{j,i}\}_{i=1}^n \in \mathbb{Z}_N$ and random elements $Q_1, Q_2, \{Q_{3,i}, Q_{4,i}\}_{i=1}^n \in \mathbb{G}_{p_2}$, $R_1, R_2, \{R_{3,i},\ R_{4,i}\}_{i=1}^n \in \mathbb{G}_{p_3}$. If $j = 1$, then it outputs a ciphertext

$$CT_{1,T} = \left(C_1 = g^{t_1} Q_1,\ C_2 = \left(u_2^T h_2\right)^{t_1} Q_2,\ \left\{D_{1,i} = \left(g^{x_{1,i}} W_{1,i}\right)^{t_1} \left(u_1^T h_1\right)^{r_{1,i}} Q_{3,i},\ D_{2,i} = g^{-r_{1,i}} Q_{4,i}\right\}_{i=1}^n \right).$$

Otherwise ($j = 2$), it outputs a ciphertext

$$CT_{2,T} = \left(E_1 = g^{t_2} R_1,\ E_2 = \left(u_1^T h_1\right)^{t_2} R_2,\ \left\{F_{1,i} = \left(g^{x_{2,i}} W_{2,i}\right)^{t_2} \left(u_2^T h_2\right)^{r_{2,i}} R_{3,i},\ F_{2,i} = g^{-r_{2,i}} R_{4,i}\right\}_{i=1}^n \right).$$

TC-FE.Decrypt$(CT_{1,T}, CT_{2,T'}, SK_{\vec{y}}, PP)$**:** This algorithm takes as input two ciphertexts $CT_{1,T} = (C_1, C_2, \{D_{1,i}, D_{2,i}\}_{i=1}^n)$ for a vector $\vec{x}_1 = (x_{1,1}, \ldots, x_{1,n})$ and $CT_{2,T'} = (E_1, E_2, \{F_{1,i}, F_{2,i}\}_{i=1}^n)$ for a vector $\vec{x}_2 = (x_{2,1}, \ldots, x_{2,n})$, a private key $SK_{\vec{y}} = K$ for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$, and the public parameters PP. If $T \neq T'$, then it outputs $\perp$. Next, it computes the following two components

$$A = e(C_1, E_1) \quad \text{and} \quad B = \prod_{i=1}^{n} \left(e(E_1, D_{1,i}) \cdot e(E_2, D_{2,i})\right)^{y_{1,i}} \cdot \prod_{i=1}^{n} \left(e(C_1, F_{1,i}) \cdot e(C_2, F_{2,i})\right)^{y_{2,i}} \cdot A^{-K}.$$

It outputs $\langle \vec{x}, \vec{y} \rangle = \sum_{j=1}^{2} \sum_{i=1}^{n} x_{j,i} \cdot y_{j,i}$ by solving the discrete logarithm of B with a base A.

Correctness. We show that the TC-FE scheme satisfies the correctness property defined in Definition 3.1. We first have $A = e(g,g)^{t_1 t_2}$. If $T = T'$, then we can derive the following equation

$$B_{1,i} = e(E_1, D_{1,i}) \cdot e(E_2, D_{2,i}) = A^{x_{1,i}+\omega_{1,i}} \cdot e(g, u_1^T h_1)^{t_2 r_1} \cdot (u_1^T h_1, g)^{-t_2 r_1} = A^{x_{1,i}+\omega_{1,i}},$$
$$B_{2,i} = e(C_1, F_{1,i}) \cdot e(C_2, F_{2,i}) = A^{x_{2,i}+\omega_{2,i}} \cdot e(g, u_2^T h_2)^{t_1 r_2} \cdot (u_2^T h_2, g)^{-t_1 r_2} = A^{x_{2,i}+\omega_{2,i}}.$$

By using the above equations, we can easily obtain the following equation

$$B = \prod_{i=1}^{n} B_{1,i}^{y_{1,i}} \cdot \prod_{i=1}^{n} B_{2,i}^{y_{2,i}} \cdot A^{-K} = \prod_{i=1}^{n} \left(A^{x_{1,i}+\omega_{1,i}}\right)^{y_{1,i}} \cdot \prod_{i=1}^{n} \left(A^{x_{2,i}+\omega_{2,i}}\right)^{y_{2,i}} \cdot A^{-\sum_{j=1}^{2}\sum_{i=1}^{n} \omega_{j,i} y_{j,i}}$$
$$= A^{\sum_{j=1}^{2}\sum_{i=1}^{n}(x_{j,i}y_{j,i}+\omega_{j,i}y_{j,i})} \cdot A^{-\sum_{j=1}^{2}\sum_{i=1}^{n} \omega_{j,i} y_{j,i}} = A^{\sum_{j=1}^{2}\sum_{i=1}^{n} x_{j,i} y_{j,i}} = A^{\langle \vec{x}, \vec{y} \rangle}.$$

3.3 Security Analysis

Theorem 3.1. *The TC-FE scheme for inner products is 1-SE-IND-secure if the C2DH and 3DH assumptions hold.*

Proof. To prove this theorem, we also consider a sequence of hybrid games $\mathbf{G}_0$ and $\mathbf{G}_F$ that are the same as those in Theorem 2.1. That is, the challenge ciphertext in $\mathbf{G}_0$ is the encryption of $(\vec{x}_1^0, \vec{x}_2^0)$ and the challenge ciphertext in $\mathbf{G}_F$ is the encryption of $(\vec{x}_1^1, \vec{x}_2^1)$. To argue the indistinguishability of hybrid games $\mathbf{G}_0$ and $\mathbf{G}_F$, we also divide the behavior of adversaries as three types depending on the size of PI. That is, an adversary is Type-0 if $|PI| = 0$, Type-1 if $|PI| = 1$, and Type-2 if $|PI| = 2$. Let $\mathbf{Adv}_{\mathcal{A}}^{G_i}$ be the advantage of $\mathcal{A}$ in a game G_i. From the Lemmas 3.2, 3.2, and 3.2, we have $\left|\mathbf{Adv}_{\mathcal{A}_0}^{G_0} - \mathbf{Adv}_{\mathcal{A}_0}^{G_F}\right| \leq \left|\mathbf{Adv}_{\mathcal{A}_0}^{G_0} - \mathbf{Adv}_{\mathcal{A}_0}^{G_F}\right| + \left|\mathbf{Adv}_{\mathcal{A}_1}^{G_0} - \mathbf{Adv}_{\mathcal{A}_1}^{G_F}\right| + \left|\mathbf{Adv}_{\mathcal{A}_2}^{G_0} - \mathbf{Adv}_{\mathcal{A}_2}^{G_F}\right| \leq 2\mathbf{Adv}_{\mathcal{B}}^{3DH}(1^\lambda) + 4\mathbf{Adv}_{\mathcal{B}}^{C2DH}(1^\lambda)$. □

3.3.1 Type-0 Adversary

Lemma 3.2. *If the 3DH assumption holds, then no polynomial-time Type-0 adversary can distinguish between $\boldsymbol{G}_0$ and $\boldsymbol{G}_F$ with a non-negligible advantage.*

Proof. The proof of this lemma for TC-FE is similar to that of Lemma 2.2 for TI-FE. That is, we also define hybrid games $\mathbf{H}_0 = \mathbf{G}_0, \mathbf{H}_1, \mathbf{H}_1'$, and $\mathbf{H}_2 = \mathbf{G}_F$ that are the same as those of Lemma 2.2. From the Lemmas 3.3, 3.4, and 3.5, we have $\left|\mathbf{Adv}_{\mathcal{A}_0}^{G_0} - \mathbf{Adv}_{\mathcal{A}_0}^{G_F}\right| \leq 2\mathbf{Adv}_{\mathcal{B}}^{3DH}(1^\lambda)$. □

Lemma 3.3. *If the 3DH assumption holds, then no polynomial-time Type-0 adversary can distinguish $\boldsymbol{H}_0$ from $\boldsymbol{H}_1$ with a non-negligible advantage.*

Proof. Suppose there exists an adversary $\mathcal{A}_0$ that breaks the security game with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the 3DH assumption using $\mathcal{A}_0$ is given: a challenge tuple $D = \big((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_2}, g_{p_3}, g_{p_1}^a, g_{p_1}^b, g_{p_1}^c\big)$ and T where $T = g_{p_1}^{abc}$ or $T = g_{p_1}^{(ab+r)c}$. Then $\mathcal{B}$ interacts with $\mathcal{A}_0$ as follows:

Init: $\mathcal{A}$ initially submits two index sets $SI = \{1,2\}$ and $PI = \emptyset$, two challenge vectors $\vec{x}^0 = (\vec{x}_1^0, \vec{x}_2^0)$ and $\vec{x}^1 = (\vec{x}_1^1, \vec{x}_2^1)$, and a challenge time period T^* where $\vec{x}_j^c = (x_{j,1}^c, \ldots, x_{j,n}^c)$.

Setup: $\mathcal{B}$ first selects random $\{w_{1,i}', w_{2,i}'\}_{i=1}^n \in \mathbb{Z}_N$ and implicitly sets the master key $\big\{\omega_{1,i} = ab(x_{1,i}^1 - x_{1,i}^0) + w_{1,i}', \omega_{2,i} = ab(x_{2,i}^1 - x_{2,i}^0) + w_{2,i}'\big\}_{i=1}^n$. It selects random $u_1', h_1', u_2', h_2' \in \mathbb{Z}_N$ and creates public parameters

$$PP = \Big((N, \mathbb{G}, \mathbb{G}_T, e),\ g = g_{p_1}, Q = g_{p_2}, R = g_{p_3},\ u_1 = g_{p_1}^a g_{p_1}^{u_1'},\ h_1 = \left(g_{p_1}^a\right)^{-T^*} g_{p_1}^{h_1'},$$
$$u_2 = g_{p_1}^a g_{p_1}^{u_2'},\ h_2 = \left(g_{p_1}^a\right)^{-T^*} g_{p_1}^{h_2'}\Big).$$

Note that it cannot create encryption keys since $g_{p_1}^{ab}$ is not given.

Query 1: $\mathcal{A}$ may adaptively request private key and ciphertext queries. $\mathcal{B}$ handles these queries as follows:

- If this is a private key query for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction $\langle \vec{x}^1 - \vec{x}^0, \vec{y} \rangle = 0$, then it creates the private key $SK_{\vec{y}} = \big(K = \sum_{i=1}^{n} w'_{1,i} y_{1,i} + \sum_{i=1}^{n} w'_{2,i} y_{2,i}\big)$.
- If this is a ciphertext query for an index $j \in \{1,2\}$, a vector $\vec{x}_j$, and a time period T with the restriction $T \neq T^*$, then it selects random $t_j, \{r'_{j,i}\}_{i=1}^{n} \in \mathbb{Z}_N$, $Q_1, Q_2, \{Q_{3,i}, Q_{4,i}\}_{i=1}^{n} \in \mathbb{G}_{p_2}$, $R_1, R_2, \{R_{3,i}, R_{4,i}\}_{i=1}^{n} \in \mathbb{G}_{p_3}$ and defines $\Delta_{j,i} = (x_{j,i}^1 - x_{j,i}^0)/(T - T^*)$. If $j = 1$, then it implicitly sets $\{r_{1,i} = -b\Delta_{1,i} t_1 + r'_{1,i}\}_{i=1}^{n}$ and creates the ciphertext

$$CT_{1,T} = \Big(C_1 = g^{t_1} Q_1,\ C_2 = \big(u_2^T h_2\big)^{t_1} Q_2,$$
$$\big\{D_{1,i} = \big(g^{x_{1,i}} g^{w'_{1,i}}\big)^{t_1} \big(g_{p_1}^b\big)^{-\Delta_{1,i} \cdot t_1 \cdot (u_1' T + h_1')} \big(u_1^T h_1\big)^{r'_{1,i}} Q_{3,i},\ D_{2,i} = \big(g_{p_1}^b\big)^{\Delta_{1,i} \cdot t_1} g^{-r'_{1,i}} Q_{4,i}\big\}_{i=1}^{n}\Big).$$

Otherwise ($j = 2$), then it implicitly sets $\{r_{2,i} = -b\Delta_{2,i} t_2 + r'_{2,i}\}_{i=1}^{n}$ and creates the ciphertext

$$CT_{2,T} = \Big(E_1 = g^{t_2} R_1,\ E_2 = \big(u_1^T h_1\big)^{t_2} R_2,$$
$$\big\{F_{1,i} = \big(g^{x_{2,i}} g^{w'_{2,i}}\big)^{t_2} \big(g_{p_1}^b\big)^{-\Delta_{2,i} \cdot t_2 \cdot (u_2' T + h_2')} \big(u_2^T h_2\big)^{r'_{2,i}} R_{3,i},\ F_{2,i} = \big(g_{p_1}^b\big)^{\Delta_{2,i} \cdot t_2} g^{-r'_{2,i}} R_{4,i}\big\}_{i=1}^{n}\Big).$$

Challenge: In the challenge step, $\mathcal{B}$ first selects random $d, \{r_{1,i}, r_{2,i}\}_{i=1}^{n} \in \mathbb{Z}_N$, $Q_1, Q_2, \{Q_{3,i}, Q_{4,i}\} \in \mathbb{G}_{p_2}$, R_1, R_2, $\{R_{3,i}, R_{4,i}\} \in \mathbb{G}_{p_3}$. Next, it implicitly sets $t_1 = c, t_2 = cd$ and creates challenge ciphertexts

$$CT_{1,T^*}^* = \Big(C_1 = g_{p_1}^c Q_1,\ C_2 = \big(g_{p_1}^c\big)^{u_2' T^* + h_2'} Q_2,$$
$$\big\{D_{1,i} = \big(g_{p_1}^c\big)^{x_{1,i}^0} (T)^{(x_{1,i}^1 - x_{1,i}^0)} \big(g_{p_1}^c\big)^{w'_{1,i}} \big(u_1^{T^*} h_1\big)^{r_{1,i}} Q_{3,i},\ D_{2,i} = g^{-r_{1,i}} Q_{4,i}\big\}_{i=1}^{n}\Big),$$
$$CT_{2,T^*}^* = \Big(E_1 = (g_{p_1}^c)^d R_1,\ E_2 = \big((g_{p_1}^c)^d\big)^{u_1' T^* + h_1'} R_2,$$
$$\big\{F_{1,i} = \big((g_{p_1}^c)^d\big)^{x^*_{(0),2,i}} (T^d)^{(x_{2,i}^1 - x_{2,i}^0)} \big((g_{p_1}^c)^d\big)^{w'_{2,i}} \big(u_2^{T^*} h_2\big)^{r_{2,i}} R_{3,i},\ F_{2,i} = g^{-r_{2,i}} R_{4,i}\big\}_{i=1}^{n}\Big).$$

Query 2: Same as Query 1.

Guess: Finally, $\mathcal{A}_0$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

Lemma 3.4. *No Type-0 adversary can distinguish $\boldsymbol{H}_1$ from $\boldsymbol{H}'_1$.*

Lemma 3.5. *If the 3DH assumption holds, then no polynomial-time Type-0 adversary can distinguish between $\boldsymbol{H}'_1$ and $\boldsymbol{H}_2$ with a non-negligible advantage.*

The proof of Lemma 3.4 is the same as that of Lemma 2.4 and the proof of Lemma 3.5 is symmetric to that of Lemma 3.3. We omit the proofs of these lemmas.

3.3.2 Type-1 and Type-2 Adversaries

Lemma 3.6. *If the C2DH assumption in $\mathbb{G}_{p_1 p_2}$ and the C2DH assumption in $\mathbb{G}_{p_1 p_3}$ hold, then no polynomial-time Type-1 adversary can distinguish between $\boldsymbol{G}_0$ and $\boldsymbol{G}_F$ with a non-negligible advantage.*

Proof. The overall strategy for the proof of this lemma is almost similar to that of Lemma 2.6 in TI-FE. Thus, we also define hybrid games $\mathbf{I}_0 = \mathbf{G}_0, \mathbf{I}_1, \mathbf{I}'_1, \mathbf{I}_2, \mathbf{I}_3, \mathbf{I}'_3$, and $\mathbf{I}_4 = \mathbf{G}_F$ that are the same as those in Lemma 2.6. From the Lemmas 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12, we have $\left|\mathbf{Adv}^{G_0}_{\mathcal{A}_1} - \mathbf{Adv}^{G_F}_{\mathcal{A}_1}\right| \leq 4\mathbf{Adv}^{C2DH}_{\mathcal{B}}(1^\lambda)$. □

Lemma 3.7. *If the C2DH assumption in $\mathbb{G}_{p_1p_2}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}_0$ from $\boldsymbol{I}_1$ with a non-negligible advantage.*

Proof. The proof of this lemma is very similar to that of Lemma 2.7 except that the public parameters additional contains some groups elements to handle time periods and ciphertexts also contains additional group elements.

Suppose there exists an adversary $\mathcal{A}$ that breaks the security game with a non-negligible advantage. A simulator $\mathcal{B}$ that solves the C2DH assumption in $\mathbb{G}_{p_1p_2}$ using $\mathcal{A}$ is given: a challenge tuple $D = \big((N, \mathbb{G}, \mathbb{G}_T, e), g_{p_1}, g_{p_2}, g_{p_3}, g^a_{p_1}Q_1, g^b_{p_1}Q_2\big)$ and T where $T = g^{ab}_{p_1}Q_3$ or $T = g^{(a+r)b}_{p_1}Q_3$. Then $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

Init: $\mathcal{A}$ submits two index sets SI and PI, two challenge vectors $\vec{x}^0 = (\vec{x}^0_1, \vec{x}^0_2)$ and $\vec{x}^1 = (\vec{x}^1_1, \vec{x}^1_2)$, and a challenge time period T^* where $|SI| = |PI| = 1$, and $\vec{x}^c_j = (x^c_{j,1}, \ldots, x^c_{j,n})$.

Setup: $\mathcal{B}$ first selects random $\{w'_{1,i}, \omega_{2,i}\}^n_{i=1} \in \mathbb{Z}_N$, $\{Q_i\}^n_{i=1} \in \mathbb{G}_{p_2}$, $\{R_i\}^n_{i=1} \in \mathbb{G}_{p_3}$ and creates two encryption keys $EK_1 = \big(\{W_{1,i} = \big(g^a_{p_1}Q_1\big)^{(x^1_{1,i} - x^0_{1,i})} g^{w'_{1,i}}_{p_1} Q_i\}^n_{i=1}\big)$ and $EK_2 = \big(\{W_{2,i} = g^{\omega_{2,i}}_{p_1} R_i\}^n_{i=1}\big)$ by implicitly setting $\big\{\omega_{1,i} = a(x^1_{1,i} - x^0_{1,i}) + w'_{1,i}\big\}^n_{i=1}$. Next, it selects random $u'_1, h'_1, u'_2, h'_2 \in \mathbb{G}_{p_1}$ and creates public parameters $PP = \big((N, \mathbb{G}, \mathbb{G}_T, e), g = g_{p_1}, Q = g_{p_2}, R = g_{p_3}, u_1 = g^{u'_1}_{p_1}, h_1 = g^{h'_1}_{p_1}, u_2 = g^{u'_2}_{p_1}, h_2 = g^{h'_2}_{p_1}\big)$. It gives $\{EK_j\}_{j \in PI}$ and PP to $\mathcal{A}$.

Query 1: $\mathcal{A}$ may adaptively request private key and ciphertext queries. If this is a private key query for a vector $\vec{y} = (\vec{y}_1, \vec{y}_2)$ with the restriction $\langle \vec{x}^1_j - \vec{x}^0_j, \vec{y}_j \rangle = 0$ for $j \in \{1, 2\}$, then it creates the private key $SK_{\vec{y}} = \big(K = \sum^n_{i=1} w'_{1,i} y_{1,i} + \sum^n_{i=1} \omega_{2,i} y_{2,i}\big)$. If this is a ciphertext query for an index $j = 2$, a vector $\vec{x}_2$, and a time period T, then it creates $CT_{2,T}$ by running $\mathbf{Encrypt}(2, \vec{x}_2, T, EK_2, PP)$ since it knows EK_2.

Challenge: In the challenge step, $\mathcal{B}$ first creates CT^*_{2,T^*} by running $\mathbf{Encrypt}(2, \vec{x}^0_2, T^*, EK_2, PP)$ since it knows $\{\omega_{2,i}\}$. Next, it selects random $\{r_{1,i}\}^n_{i=1} \in \mathbb{Z}_N$, $Q'_1, Q'_2, \{Q'_{3,i}, Q'_{4,i}\}^n_{i=1} \in \mathbb{G}_{p_2}$ and creates challenge ciphertext by implicitly setting $t_1 = b$ as

$$CT^*_{1,T^*} = \Big(C_1 = \big(g^b_{p_1}Q_2\big)Q'_1,\ C_2 = \big(g^b_{p_1}Q_2\big)^{u'_2 T^* + h'_2} Q'_2,$$
$$\big\{D_{1,i} = \big(g^b_{p_1}Q_2\big)^{x^0_{1,i}} (T)^{(x^1_{1,i} - x^0_{1,i})} \big(g^b_{p_1}Q_2\big)^{w'_{1,i}} \big(u^{T^*}_1 h_1\big)^{r_{1,i}} Q'_{3,i},\ D_{2,i} = g^{-r_{1,i}} Q'_{4,i}\big\}^n_{i=1}\Big).$$

Query 2: Same as Query 1.

Guess: Finally, $\mathcal{A}_1$ outputs a guess μ'. $\mathcal{B}$ also outputs μ'. □

Lemma 3.8. *No Type-1 adversary can distinguish $\boldsymbol{I}_1$ from $\boldsymbol{I}'_1$.*

Lemma 3.9. *If the C2DH assumption in $\mathbb{G}_{p_1p_2}$ holds, then no polynomial-time Type-1 adversary can distinguish between $\boldsymbol{I}'_1$ and $\boldsymbol{I}_2$ with a non-negligible advantage.*

The proof of Lemma 3.8 is the same as that of Lemma 2.8 in TI-FE. The proof of Lemma 3.9 is symmetric to that of Lemma 3.7.

Lemma 3.10. *If the C2DH assumption in $\mathbb{G}_{p_1p_3}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}_2$ from $\boldsymbol{I}_3$ with a non-negligible advantage.*

Lemma 3.11. *No Type-1 adversary can distinguish $\boldsymbol{I}_3$ from $\boldsymbol{I}'_3$.*

Lemma 3.12. *If the C2DH assumption in $\mathbb{G}_{p_1p_3}$ holds, then no polynomial-time Type-1 adversary can distinguish $\boldsymbol{I}'_3$ from $\boldsymbol{I}_4$ with a non-negligible advantage.*

The proof of Lemma 3.10 is very similar to that of Lemma 3.7 except that the C2DH assumption in $\mathbb{G}_{p_1p_3}$ is used to embed itself in the index 2. The proof of Lemma 3.11 is almost the same as that of Lemma 3.8. The proof of Lemma 3.12 is symmetric to that of Lemma 3.10. Thus, we omit the proof of this lemma.

Lemma 3.13. *If the C2DH assumption in $\mathbb{G}_{p_1p_2}$ and the C2DH assumption in $\mathbb{G}_{p_1p_3}$ hold, then no polynomial-time Type-2 adversary can distinguish between $\boldsymbol{G}_0$ and $\boldsymbol{G}_F$ with a non-negligible advantage.*

The proof of Lemma 3.13 also can be done similarly to that of Lemma 3.6. We omit the details of this proof.

4 Two-Input FE in Prime-Order Groups

In this section, we convert our TI-FE and TC-FE schemes in composite-order bilinear groups into TI-FE and TC-FE schemes in prime-order (asymmetric) bilinear groups.

4.1 Asymmetric Bilinear Groups

Let $\mathcal{G}_{as}$ be a group generator algorithm that takes as input a security parameter λ and outputs a tuple $(p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e)$ where p is a random prime and $\mathbb{G},\hat{\mathbb{G}}$, and $\mathbb{G}_T$ be three cyclic groups of prime order p. Let g and $\hat{g}$ be generators of $\mathbb{G}$ and $\hat{\mathbb{G}}$, respectively. The bilinear map $e:\mathbb{G}\times\hat{\mathbb{G}}\to\mathbb{G}_T$ has the following properties:

1. Bilinearity: $\forall u\in\mathbb{G},\forall\hat{v}\in\hat{\mathbb{G}}$ and $\forall a,b\in\mathbb{Z}_p$, $e(u^a,\hat{v}^b)=e(u,\hat{v})^{ab}$.
2. Non-degeneracy: $\exists g\in\mathbb{G},\hat{g}\in\hat{\mathbb{G}}$ such that $e(g,\hat{g})$ has order p in $\mathbb{G}_T$.

We say that $\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T$ are asymmetric bilinear groups with no efficiently computable isomorphisms if the group operations in $\mathbb{G},\hat{\mathbb{G}}$, and $\mathbb{G}_T$ as well as the bilinear map e are all efficiently computable, but there are no efficiently computable isomorphisms between $\mathbb{G}$ and $\hat{\mathbb{G}}$.

4.2 Complexity Assumptions

Assumption 3 (Symmetric External Diffie-Hellman, SXDH)**.** Let $(p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e)$ be a tuple randomly generated by $\mathcal{G}_{as}(1^\lambda)$ where p is prime order of the groups. Let $g,\hat{g}$ be random generators of groups $\mathbb{G},\hat{\mathbb{G}}$ respectively. The SXDH assumption is that the DDH assumption holds in both $\mathbb{G}$ and $\hat{\mathbb{G}}$. That is, if the challenge tuple

$$D=\left((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,\hat{g},g^a,g^b\right) \text{ and } T$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $T=T_0=g^{ab}$ from $T=T_1=g^c$ with more than a negligible advantage, and if the challenge tuple

$$D=\left((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,\hat{g},\hat{g}^a,\hat{g}^b\right) \text{ and } T$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $T=T_0=\hat{g}^{ab}$ from $T=T_1=\hat{g}^c$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{SXDH}_{\mathcal{A}}(1^\lambda)=\left|\Pr[\mathcal{A}(D,T_0)=0]-\Pr[\mathcal{A}(D,T_1)=0]\right|$ where the probability is taken over random choices of $a,b,c\in\mathbb{Z}_p$.

Assumption 4 (Asymmetric 3-Party Diffie-Hellman, a3DH)**.** Let $(p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e)$ be a tuple randomly generated by $\mathcal{G}_{as}(1^\lambda)$ where p is prime order of the groups. Let $g,\hat{g}$ be random generators of $\mathbb{G},\hat{\mathbb{G}}$ respectively. The a3DH assumption is that if the challenge tuple

$$D=\left((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,\hat{g},g^a,g^b,g^c,\hat{g}^a,\hat{g}^b,\hat{g}^c\right) \text{ and } T=(S_1,S_2)$$

are given, no PPT algorithm $\mathcal{A}$ can distinguish $T=T_0=\left(g^{abc},\hat{g}^{abc}\right)$ from $T=T_1=\left(g^{(ab+r)c},\hat{g}^{(ab+r)c}\right)$ with more than a negligible advantage. The advantage of $\mathcal{A}$ is defined as $\mathbf{Adv}^{a3DH}_{\mathcal{A}}(1^\lambda)=\left|\Pr[\mathcal{A}(D,T_0)=0]-\Pr[\mathcal{A}(D,T_1)=0]\right|$ where the probability is taken over random choices of $a,b,c_1,c_2,r\in\mathbb{Z}_p$.

4.3 TI-FE Construction

In composite-order bilinear groups, we used an additional subgroup for the randomization of group elements to prevent simple attacks that use the pairing operation. That is, the ciphertext components of CT_1, CT_2 use $Q\in\mathbb{G}_{p_2}$, $R\in\mathbb{G}_{p_3}$ for randomization respectively. To convert our TI-FE and TC-FE schemes in composite-order groups to prime-order groups, we use the asymmetry of prime-order asymmetric bilinear groups. That is, the ciphertext CT_1 consists of group elements in $\mathbb{G}$ and the ciphertext CT_2 consists of group elements in $\hat{\mathbb{G}}$. Recall that the DDH assumption holds in asymmetric bilinear groups.

TI-FE.Setup$(1^\lambda,n)$**:** It first obtains $(p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e)$ by running $\mathcal{G}_{as}(1^\lambda)$ where p is prime order of groups. Let $g,\hat{g}$ be random generators of $\mathbb{G},\hat{\mathbb{G}}$ respectively. Next, it chooses random $\{\omega_{1,i},\omega_{2,i}\}_{i=1}^n\in\mathbb{Z}_p$. It outputs a master key $MK=\left(\{\omega_{1,i},\omega_{2,i}\}_{i=1}^n\right)$, two encryption keys $EK_1=\left(\{w_{1,i}=g^{\omega_{1,i}}\}_{i=1}^n\right)$ and $EK_2=\left(\{\hat{w}_{2,i}=\hat{g}^{\omega_{2,i}}\}_{i=1}^n\right)$, and public parameters $PP=\left((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,\hat{g}\right)$.

TI-FE.GenKey$(\vec{y}=(\vec{y}_1,\vec{y}_2),MK,PP)$**:** Let $\vec{y}=(\vec{y}_1,\vec{y}_2)$ where $\vec{y}_j=(y_{j,1},\ldots,y_{j,n})$. It outputs a private key $SK_{\vec{y}}=\left(K=\sum_{j=1}^2\sum_{i=1}^n\omega_{j,i}\cdot y_{j,i}\right)\in\mathbb{Z}_p$.

TI-FE.Encrypt$(j,\vec{x}_j,EK_j,PP)$**:** It first chooses a random exponent $t_j\in\mathbb{Z}_p$. If $j=1$, then it outputs a ciphertext $CT_1=\left(C_1=g^{t_1},\left\{D_{1,i}=\left(g^{x_{1,i}}w_{1,i}\right)^{t_1}\right\}_{i=1}^n\right)\in\mathbb{G}^{n+1}$. Otherwise ($j=2$), it outputs a ciphertext $CT_2=\left(E_1=\hat{g}^{t_2},\left\{F_{1,i}=\left(\hat{g}^{x_{2,i}}\hat{w}_{2,i}\right)^{t_2}\right\}_{i=1}^n\right)\in\hat{\mathbb{G}}^{n+1}$.

TI-FE.Decrypt$(CT_1,CT_2,SK_{\vec{y}},PP)$**:** Let $CT_1=(C_1,\{D_{1,i}\}_{i=1}^n)$ for a vector $\vec{x}_1=(x_{1,1},\ldots,x_{1,n})$ and $CT_2=(E_1,\{F_{1,i}\}_{i=1}^n)$ for a vector $\vec{x}_2=(x_{2,1},\ldots,x_{2,n})$, and $SK_{\vec{y}}=(K_1)$ for a vector $\vec{y}=(\vec{y}_1,\vec{y}_2)$. It first computes $A=e(C_1,E_1)$ and $B=\prod_{i=1}^n e(D_{1,i},E_1)^{y_{1,i}}\cdot\prod_{i=1}^n e(C_1,F_{1,i})^{y_{2,i}}\cdot A^{-K}$. Next, it outputs $\langle\vec{x},\vec{y}\rangle=\sum_{j=1}^2\sum_{i=1}^n x_{j,i}\cdot y_{j,i}$ by solving the discrete logarithm of B with a base A.

4.4 TC-FE Construction

TC-FE.Setup$(1^\lambda,n)$**:** This algorithm first obtains $(p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e)$ by running $\mathcal{G}_{as}(1^\lambda)$ where p is prime order of the groups. Let $g,\hat{g}$ be random generators of the groups $\mathbb{G},\hat{\mathbb{G}}$ respectively. It chooses random exponents $\{\omega_{1,i},\omega_{2,i}\}_{i=1}^n\in\mathbb{Z}_p$. It also selects random exponents $u_1',h_1',u_2',h_2'\in\mathbb{Z}_p$. It outputs a master key $MK=\left(\{\omega_{1,i},\omega_{2,i}\}_{i=1}^n\right)$, two encryption keys $EK1=\left(\{w_{1,i}=g^{\omega_{1,i}}\}_{i=1}^n\right)$ and $EK2=\left(\{\hat{w}_{2,i}=\hat{g}^{\omega_{2,i}}\}_{i=1}^n\right)$, and public parameters $PP=\left((p,\mathbb{G},\hat{\mathbb{G}},\mathbb{G}_T,e),g,\hat{g},\{u_j=g^{u_j'},h_j=g^{h_j'},\hat{u}_j=\hat{g}^{u_j'},\hat{h}_j=\hat{g}^{h_j'}\}_{j=1}^2\right)$.

TC-FE.GenKey$(\vec{y}=(\vec{y}_1,\vec{y}_2),MK,PP)$**:** Let $\vec{y}=(\vec{y}_1,\vec{y}_2)$ where $\vec{y}_j=(y_{j,1},\ldots,y_{j,n})$. It outputs a private key $SK_{\vec{y}}=\left(K=\sum_{j=1}^2\sum_{i=1}^n\omega_{j,i}\cdot y_{j,i}\right)\in\mathbb{Z}_p$.

TC-FE.Encrypt$(j,\vec{x}_j,T,EK_j,PP)$**:** It first chooses random exponents $t_j,\{r_{j,i}\}_{i=1}^n \in \mathbb{Z}_p$. If $j=1$, then it outputs a ciphertext $CT_{1,T} = \big(C_1 = g^{t_1},\ C_2 = \big(u_2^T h_2\big)^{t_1},\ \big\{D_{1,i} = \big(g^{x_{1,i}} w_{1,i}\big)^{t_1}\big(u_1^T h_1\big)^{r_{1,i}},\ D_{2,i} = g^{-r_{1,i}}\big\}_{i=1}^n\big)$. Otherwise ($j=2$), it outputs a ciphertext $CT_{2,T} = \big(E_1 = \hat{g}^{t_2},\ E_2 = \big(\hat{u}_1^T \hat{h}_1\big)^{t_2},\ \big\{F_{1,i} = \big(\hat{g}^{x_{2,i}} \hat{w}_{2,i}\big)^{t_2}\big(\hat{u}_2^T \hat{h}_2\big)^{r_{2,i}},\ F_{2,i} = \hat{g}^{-r_{2,i}}\big\}_{i=1}^n\big)$.

TC-FE.Decrypt$(CT_{1,T},CT_{2,T'},SK_{\vec{y}},PP)$**:** Let $CT_{1,T} = (C_1,C_2,\{D_{1,i},D_{2,i}\}_{i=1}^n)$ for a vector $\vec{x}_1 = (x_{1,1},\ldots,x_{1,n})$ and $CT_{2,T'} = (E_1,E_2,\{F_{1,i},F_{2,i}\}_{i=1}^n)$ for a vector $\vec{x}_2 = (x_{2,1},\ldots,x_{2,n})$, and $SK_{\vec{y}} = K$ for a vector $\vec{y} = (\vec{y}_1,\vec{y}_2)$. If $T \neq T'$, then it outputs $\perp$. Next, it computes $A = e(C_1,E_1)$ and $B = \prod_{i=1}^n \big(e(D_{1,i},E_1)\cdot e(D_{2,i},E_2)\big)^{y_{1,i}} \cdot \prod_{i=1}^n \big(e(C_1,F_{1,i})\cdot e(C_2,F_{2,i})\big)^{y_{2,i}} \cdot A^{-K}$. It outputs $\langle\vec{x},\vec{y}\rangle = \sum_{j=1}^2 \sum_{i=1}^n x_{j,i}\cdot y_{j,i}$ by solving the discrete logarithm of B with a base A.

4.5 Security Analysis

The security proofs of our TI-FE and TC-FE schemes in asymmetric bilinear groups are almost similar to those of our TI-FE and TC-FE schemes in composite-order bilinear groups except that the SXDH and a3DH assumptions are used.

Theorem 4.1. *The above TI-FE scheme for inner products is 1-SE-IND-secure if the SXDH and a3DH assumptions hold.*

Theorem 4.2. *The above TC-FE scheme for inner products is 1-SE-IND-secure if the SXDH and a3DH assumptions hold.*

The proof of Theorem 4.1 is almost similar to that of Theorem 2.1 except that the SXDH and a3DH assumptions are used instead of the C2DH and 3DH assumptions. Note that the C2DH assumption in $\mathbb{G}_{p_1p_2}$ (or $\mathbb{G}_{p_1p_3}$) of composite-order bilinear groups is directly corresponding to the DDH assumption in $\mathbb{G}$ (or $\hat{\mathbb{G}}$) of asymmetric bilinear groups. We omit the detailed proofs. The proof of Theorem 4.2 is almost similar to that of Theorem 3.1. We also omit the detailed proof.

5 Conclusion

In this paper, we showed that TI-FE and TC-FE schemes for inner products can be built on composite-order bilinear groups and proved their selective IND-security under simple assumptions. Furthermore, we proposed TI-FE and TC-FE schemes in prime-order bilinear groups by relying on the asymmetric property of asymmetric bilinear groups. We hope that our new approach to build TI-FE schemes may provide an interesting insight to build a more practical MI-FE scheme.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 205–222. Springer, 2005.

[2] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In Jonathan Katz, editor, *Public-Key Cryptography - PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 733–751. Springer, 2015.

[3] Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Better security for functional encryption for inner product evaluations. Cryptology ePrint Archive, Report 2016/011, 2016. `http://eprint.iacr.org/2016/011`.

[4] Michel Abdalla, Mariana Raykova, and Hoeteck Wee. Multi-input inner-product functional encryption from pairings. Cryptology ePrint Archive, Report 2016/425, 2016. `http://eprint.iacr.org/2016/425`.

[5] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011*, volume 7073 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.

[6] Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016*, volume 9816 of *Lecture Notes in Computer Science*, pages 333–362. Springer, 2016.

[7] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, volume 9216 of *Lecture Notes in Computer Science*, pages 657–677. Springer, 2015.

[8] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, volume 9215 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2015.

[9] Saikrishna Badrinarayanan, Divya Gupta, Abhishek Jain, and Amit Sahai. Multi-input functional encryption for unbounded arity functions. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015*, volume 9452 of *Lecture Notes in Computer Science*, pages 27–51. Springer, 2015.

[10] Allison Bishop, Abhishek Jain, and Lucas Kowalczyk. Function-hiding inner product encryption. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015*, volume 9452 of *Lecture Notes in Computer Science*, pages 470–491. Springer, 2015.

[11] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.

[12] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer, 2004.

[13] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[14] Dan Boneh, Kevin Lewi, Mariana Raykova, Amit Sahai, Mark Zhandry, and Joe Zimmerman. Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, volume 9057 of *Lecture Notes in Computer Science*, pages 563–594. Springer, 2015.

[15] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *Theory of Cryptography - TCC 2011*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273. Springer, 2011.

[16] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

[17] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *Theory of Cryptography - TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer, 2007.

[18] Zvika Brakerski, Ilan Komargodski, and Gil Segev. Multi-input functional encryption in the private-key setting: Stronger security from weaker assumptions. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 852–880. Springer, 2016.

[19] Zvika Brakerski and Gil Segev. Function-private functional encryption in the private-key setting. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - TCC 2015*, volume 9015 of *Lecture Notes in Computer Science*, pages 306–324. Springer, 2015.

[20] Pratish Datta, Ratna Dutta, and Sourav Mukhopadhyay. Functional encryption for inner product with full function privacy. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *Public-Key Cryptography - PKC 2016*, volume 9614 of *Lecture Notes in Computer Science*, pages 164–195. Springer, 2016.

[21] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49. IEEE Computer Society, 2013.

[22] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Functional encryption without obfuscation. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - TCC 2016-A*, volume 9563 of *Lecture Notes in Computer Science*, pages 480–511. Springer, 2016.

[23] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *ACM Symposium on Theory of Computing - STOC 2009*, pages 169–178. ACM, 2009.

[24] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.

[25] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC 2013*, pages 555–564. ACM, 2013.

[26] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 162–179. Springer, 2012.

[27] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, volume 9216 of *Lecture Notes in Computer Science*, pages 503–523. Springer, 2015.

[28] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. Functional encryption for randomized functionalities. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - TCC 2015*, volume 9015 of *Lecture Notes in Computer Science*, pages 325–351. Springer, 2015.

[29] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[30] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer, 2008.

[31] Kwangsu Lee and Dong Hoon Lee. Improved hidden vector encryption with short ciphertexts and tokens. *Designs Codes Cryptogr.*, 58(3):297–319, 2011.

[32] Jong Hwan Park. Inner-product encryption under standard assumptions. *Designs Codes Cryptogr.*, 58(3):235–257, 2011.

[33] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security, CCS 2010*, pages 463–472. ACM, 2010.

[34] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *Theory of Cryptography - TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2009.

[35] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015*, volume 9216 of *Lecture Notes in Computer Science*, pages 678–697. Springer, 2015.