Embedded Identity Traceable Identity-Based IPFE from Pairings and Lattices

Subhranil Dutta[1], Tapas Pal[2], Amit Kumar Singh[1], Sourav Mukhopadhyay[1]

[1] Indian Institute of Technology Kharagpur, West Bengal, India-721302
{subhranildutta,amitsingh}@iitkgp.ac.in, sourav@maths.iitkgp.ac.in
[2] NTT Social Informatics Laboratories
tapas.pal.wh@hco.ntt.co.jp

Abstract.

We present the *first* fully collusion resistant traitor tracing (TT) scheme for identity-based inner product functional encryption (IBIPFE) that directly traces user identities through an efficient tracing procedure. We name such a scheme as *embedded identity traceable IBIPFE* (EI-TIBIPFE), where secret keys and ciphertexts are computed for vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ respectively. Additionally, each secret key is associated with a user identification information tuple $(i, \mathsf{id}, \mathsf{gid})$ that specifies user index i, user identity id and an identity gid of a group to which the user belongs. The ciphertexts are generated under a group identity gid' so that decryption recovers the inner product between the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ if the user is a member of the group gid', i.e., $\mathsf{gid} = \mathsf{gid}'$. Suppose some users linked to a particular group team up and create a pirate decoder that is capable of decrypting the content of the group, then the tracing algorithm extracts at least one id from the team given black-box access to the decoder.

In prior works, such TT schemes are built for usual public key encryptions. The only existing TIPFE scheme proposed by Do, Phan, and Pointcheval [CT-RSA'20] can trace user indices but not the actual identities. Moreover, their scheme achieves *selective* security and *private* traceability, meaning that it is only the trusted authority that is able to trace user indices. In this work, we present the following TT schemes with varying parameters and levels of security:

(1) We generically construct EI-TIBIPFE assuming the existence of IBIPFE. The scheme preserves the security level of the underlying IBIPFE.
(2) We build an adaptively secure EI-TIPFE scheme from bilinear maps. Note that EI-TIPFE is a particular case of EI-TIBIPFE, which does not consider group identities.
(3) Next, we construct a selectively secure EI-TIBIPFE from bilinear maps. As an intermediate step, we design the *first* IBIPFE scheme based on a target group assumption in the standard model.
(4) Finally, we provide a generic construction of selectively secure EI-TIBIPFE from lattices, namely under the standard Learning With Errors assumption.

Our pairing-based schemes support *public traceability* and the ciphertext size grows with $\sqrt{n}$, whereas in the IBIPFE and lattice-based ones, it grows linearly with n. The main technical difficulty is designing such an advanced TT scheme for an IBIPFE that is beyond IPFE and more suitable for real-life applications.

Keywords: embedded identity, traitor tracing, inner product functional encryption, identity-based inner product functional encryption

Table of Contents

1 Introduction

A traditional *traitor tracing* (TT) [CFN94] scheme is a multi-receiver system that helps to detect a malicious user that deceives the broadcasters by creating a pirate decryption box. More specifically, contents are encrypted under a public key mpk and each authorized user indexed with j is given a sophisticated secret key sk_j to recover the contents. Consider a scenario where a collection of dishonest users, called *traitors*, embeds their secret keys into a pirate decoder which decrypts the ciphertext for unauthorized users, thereby causing a significant loss to the content providers. To prevent such impermissible theft, there is a tracing algorithm that uses a dedicated tracing key key to identify the traitors in the system. The tracing algorithm is called *public* or *private*, depending on whether the key is available publicly or kept secret.

In literature, TT schemes are mainly explored in the context of usual *public key encryption* (PKE) [CFN94,BF99,SW98,KD98,CFNP00,FT01,SSW01,CPP05,BF99,TT01,KY02a,KY02b, FNP07, BP08, BZ17, LPSS17, ABP+17] or *identity-based encryption* (IBE) [ADML+07, PT11, GMS12]. Recently, Do, Phan and Pointcheval [DPP20] bring this feature of traceability into the setting of a more fine-grained encryption mechanism called *functional encryption* (FE) [BSW11]. In particular, they define and construct a *traceable inner product functional encryption* (TIPFE) scheme where secret keys are generated for tuples $(j, \boldsymbol{u})$ representing user indices and vectors. The ciphertexts are computed for some vectors $\boldsymbol{v}$ in such a manner that the decryption reveals nothing about the message $\boldsymbol{v}$ except the inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$. Suppose many secret keys $\mathsf{sk}_{j,\boldsymbol{u}}$ for a fixed vector $\boldsymbol{u}$ are provided to different users having distinct indices. It may happen that some of these users create a pirate decoder embedding their own secret keys in order to sell it for personal interests. Therefore, anyone from outside can learn the inner product using the pirate decoder. The tracing algorithm of TIPFE is employed to identify such dishonest users in the system.

Following the usual tracing procedures [BSW06,BF99], Do et al. [DPP20] design their tracing algorithm to find out a set T^{index} containing the traitor's indices associated with the secret keys of IPFE. In order to find the actual traitors, the indices in T^{index} are *mapped back* to the identities of traitors. Thus, the central authority must maintain a *map* or a look-up table to discover the identities linked to the indices of T^{index}. Additionally, the key generation process of [DPP20] encodes the identities as codewords or vectors (having the same length as the IPFE vectors), and the tracing algorithm needs to access the list of these codewords. This makes the key generation inherently *stateful*, which not only dilutes the whole purpose of tracing but is inconvenient for many practical scenarios. Further, a TIPFE like [DPP20] which supports *private* tracing is more restrictive since *only* the central authority can find out the traitors' identities. On the other hand, even if the *map* is made publicly available, then users' anonymity would be fully compromised.

Nishimaki, Wichs and Zhandry [NWZ16] address this issue by constructing a TT scheme that directly traces users' identities publicly from a decoder box. However, their TT scheme is built upon an adaptively-secure collusion-resistant public key FE scheme for general circuits with compact ciphertexts. All known constructions of such FE schemes depend on the existence of either multilinear maps [GGH+13] or indistinguishability obfuscation (IO) [BV16]. Although IO has been built from well-founded assumptions [JLS21] through a long sequence of works, the construction is highly complex and relies on various cryptographic tools. Nevertheless, it has not yet reached the arena of implementable primitives. To avoid the route of full-fledge FE or IO, Goyal et al. [GKW19] developed new and more efficient public/private fully collusion resistant traitor tracing schemes from standard assumptions based on paring and lattices. However, these TT schemes can trace users' identities *only* in plain PKEs. The

only existing TIPFE of [DPP20] is proven secure in the *selective* indistinguishability model and achieves *private* tracing (that can trace only user indices) under the *decisional bilinear Diffie-Hellman* (DBDH) assumption. Hence, there is no known technique for implementing such an advanced identity tracing mechanism in the case of IPFE with a realistic security model. This motivates us to the following question:

Is it possible to construct an efficient, adaptively secure fully collusion resistant TIPFE scheme where the user identification information can be embedded into the secret keys such that the tracing algorithm publicly traces the indices associated with traitors' secret keys as well as their identities?

We further investigate more realistic applications of TIPFE. In the current structure of TIPFE, the ciphertexts do not contain any information about the source of the message. As a result, the sender can not be recognized during decryption, although it is an essential and desirable feature for any PKE scheme. To capture this phenomenon, consider a natural scenario where many broadcasters (or data suppliers) encrypt their data using the same TIPFE. Then, users having secret keys from one broadcaster can decrypt the content of others. Further, in the context of tracing, this means the tracing must be run across all the users' identities even when the decoder is created to cheat a specific broadcaster. A naive way to resolve this problem is to sample individual TIPFE system for each of these broadcasters. However, this requires maintaining a huge database for storing parameters of all the TIPFE systems, and complications may arise in managing certificates. Such shortcomings of TIPFE can be surpassed by introducing identities assigned to the broadcasters and enabling them to encrypt contents under their own individual identities. A secret key is now associated with an additional broadcaster's identity, which restricts the user to decrypt only the content of that broadcaster. This is motivated by the notion of an *identity-based* TT (IB-TT) scheme [ADML+07]. However, all existing IB-TTs can only trace user indices (but not their identities), and hence possesses similar limitations regarding tracing as in the TIPFE of [DPP20]. Hence, in continuation with the above open problem, we are further motivated to answer the following question:

Is it possible to construct an efficient fully collusion resistant TIPFE scheme where a user's secret key is additionally associated with the subscribed broadcaster's identity, and messages can be encrypted under broadcasters' identities such that only its' subscribed users can view the content while the tracing algorithm privately/publicly traces the traitors' indices along with their identities?

Our Contributions. In this work, we affirmatively answer to the above questions. More precisely, we make the following contributions.

Embedded Identity TIPFE. We formally introduce the notion of *embedded identity* TIPFE (EI-TIPFE) inspired from the primitive of EI-TT introduced in [GKW19]. We further extend this notion to *identity-based* IPFE (IBIPFE) [ACGU20] and define a more generalized primitive called *embedded identity traceable* IBIPFE (EI-TIBIPFE). It additionally encodes broadcaster's identity, referred to as *group identity* here after, into a secret key and enables encrypting message vectors under group identities. We emphasize that only user identities are traced in EI-TIBIPFE from a pirate decoder designed for a specific group identity.

To construct EI-TIPFE or EI-TIBIPFE, we formalize an intermediate primitive called *embedded identity private linear* IBIPFE (EIPL-IBIPFE). Also, we observe that EIPL-IPFE can be viewed as a particular case of it where the group identities are omitted. We show that

EI-TIBIPFE or EI-TIPFE can be achieved generically from EIPL-IBIPFE or EIPL-IPFE respectively. Hence, the answer to the above questions boils down to construct the intermediate primitive EIPL-IBIPFE.

EIPL-IBIPFE from IBIPFE. We present a generic construction of EIPL-IBIPFE from IBIPFE, which preserves the security of the underlying IBIPFE. It can publicly trace user identities from the pirate decoder. The ciphertext size and length of the public key grow linearly in both the numbers of users n and the length of embedded user identities k.

EIPL-IPFE from pairing. We propose an *adaptively* secure EIPL-IPFE in a composite order pairing group based on a DBDH-like assumption called the *joint Decisional 3-party Diffie-Hellman* (joint-D3DH) assumption. It supports public tracing with more efficient system parameters. Specifically, the size of ciphertexts and the master public keys grow linearly with $\sqrt{n}, \sqrt{k}$.

EIPL-IBIPFE from pairing. We propose a *selectively* secure EIPL-IBIPFE in a composite-order pairing group based on the joint-D3DH assumption. We follow a two steps approach – in the first step, we build an IBIPFE scheme based on the (plain) *Decisional Bilinear Diffie-Hellman* (DBDH) assumption, and in the second step, we upgrade the IBIPFE to EIPL-IBIPFE that also supports public traceability. Moreover, the EIPL-IBIPFE enjoys similar efficiency as our EIPL-IPFE.

We note that our IBIPFE is built upon a target group assumption, namely DBDH, in the standard model. All existing group-based IBIPFEs are either relying on source group assumptions [ACGU20, AGT21] or built in the random oracle model [DSP19]. It is well-known that target group assumptions are qualitatively weaker than the source group ones [Fre10, DKW21]. Hence, it is worth mentioning that our IBIPFE is the *first* instantiation of a FE scheme beyond IPFE that is designed from a simple target group assumption in the standard model.

EIPL-IBIPFE from lattices. Lastly, we propose a generic construction of selectively secure EIPL-IBIPFE assuming the existence of *mixed* FE (MFE) and *attribute-based* IPFE (ABIPFE). Both of these primitives are known from the learning with error (LWE) assumption. We instantiate our EIPL-IBIPFE using the ABIPFE of [LLW21] and the MFE of [CVW+18]. This yields a post-quantum secure EI-TIBIPFE with ciphertext size $m \cdot (n + k + k') \cdot \mathsf{poly}(\lambda)$ and public key size $m \cdot \mathsf{poly}(\lambda)$.

Like [BSW06, Fre10, BZ17, BW06, GKRW18, CVW+18, NWZ16, GKW19], our EI-TIPFE or EI-TIBIPFE schemes are fully collusion resistant meaning that there is no bound on adversary's secret key queries. In particular, for our lattice-based EI-TIBIPFE scheme, the collusion bound is set to be the total number of users in the system, which is similar to the collusion resistant setting of [NWZ16].

The joint-D3DH assumption. Our joint-D3DH assumption is a natural generalization of the commonly used D3DH assumption [BSW06, GKW19]. Let $\mathbb{G}$ be a cyclic group of order $N = p \cdot q$ with subgroups $\mathbb{G}_p$, $\mathbb{G}_q$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an efficiently computable bilinear map. The D3DH assumption is stated as follows:

$$g \in \mathbb{G};\ a, b, c, r \leftarrow \mathbb{Z}_N :\ \{g^a, g^b, g^c, g^{abc}\} \approx \{g^a, g^b, g^c, g^r\}$$

Let g_p, g_q are two generators of $\mathbb{G}_p$ and $\mathbb{G}_q$ respectively. Inspired from the joint-*symmetric external Diffie-Hellman* (joint-SXDH) assumption introduced by Lin et al. [LV16], we now

generalize the D3DH assumption. More specifically, the joint-D3DH assumption considers the joint distribution of elements $\{g_i^a, g_i^b, g_i^c\}_{i\in\{p,q\}}$ with the same exponents $a,b,c \leftarrow \mathbb{Z}_N$. Note that, in the symmetric pairing groups, elements of two subgroups $\mathbb{G}_p$ and $\mathbb{G}_q$ are *not pairable* in the sense that $e(g_p^a, g_q^b) = 1$ for any a,b. Therefore, by the same spirit of D3DH, the distribution of $\{g_p^{abc}, g_q^{abc}\}$ is possibly indistinguishable from $\{g_p^r, g_q^r\}$ for $r \leftarrow \mathbb{Z}_N$ – this is exactly our joint-D3DH assumption. We state the assumption as follows:

$$g_p \in \mathbb{G}_p, g_q \in \mathbb{G}_q;\ a,b,c,r \leftarrow \mathbb{Z}_N : \left\{ \begin{matrix} g_p^a,\ g_p^b,\ g_p^c,\ g_p^{abc}, \\ g_q^a,\ g_q^b,\ g_q^c,\ g_q^{abc} \end{matrix} \right\} \approx \left\{ \begin{matrix} g_p^a,\ g_p^b,\ g_p^c,\ g_p^r, \\ g_q^a,\ g_q^b,\ g_q^c,\ g_q^r \end{matrix} \right\}$$

Table 1: Comparison between Traceable FEs

Scheme	Assum.	Size of the component			Tracing mode	(Func., Sec.)	Identity trace
		|mpk|	|ct|	|sk|			
[DPP20]	DBDH	$m(n+\text{poly}(\lambda))$	$m\cdot\text{poly}(\lambda)$	$\text{poly}(\lambda)$	Private	(IPFE, Sel)	×
Our work	Joint- D3DH	$m\cdot\sqrt{n\cdot k}\cdot\text{poly}(\lambda)$	$m\cdot\sqrt{n\cdot k}\cdot\text{poly}(\lambda)$	$\log n+\text{poly}(\lambda)$	Public	(IPFE, Adp)	✓
	IBIPFE	$m\cdot n\cdot k\cdot\text{poly}(\lambda)$	$m\cdot n\cdot k\cdot\text{poly}(\lambda)$	$k'+k\cdot\text{poly}(\lambda)$	Public	(IBIPFE, Adp)	✓
	Joint-D3DH	$m\cdot\sqrt{n\cdot k}\cdot\text{poly}(\lambda)$	$m\cdot\sqrt{n\cdot k}\cdot\text{poly}(\lambda)$	$\log n+\text{poly}(\lambda)$	Public	(IBIPFE, Sel)	✓
	LWE[†]	$m\cdot\text{poly}(\lambda)$	$m\cdot(n+k+k')\cdot\text{poly}(\lambda)$	$(\log n+k+k')\cdot\text{poly}(\lambda)$	Private	(IBIPFE, Sel)	✓

n: number of user; m, k, k': dimension of input vector, user identity and group identity respectively; Func., Sec.: functionality and security model; Sel, Adp: selective, adaptive respectively. [†] We instantiate our LWE-based EI-TIBIPFE using the MFE scheme of [GKW18] and the ABIPFE scheme of [LLW21]. Therefore, the parameters of our LWE-based scheme are directly aligned to [GKW18, LLW21].

Finally, we summarize our results in the following Theorem.

Theorem 1 (Informal) *Assuming IBIPFE/joint-D3DH/LWE assumption, there exist an adaptively/selectively(adaptively)/selectively secure EI-TIBIPFE (EI-TIPFE) scheme with public/ public/private tracing algorithm that can trace user identities from a pirate decoder.*

Recently, Zhandry [Zha21] proposed white-box tracing mechanism for PKE which allows the tracer to inspect the implementation of decoder and prevents some attack scenarios that are inherent to black-box traitor tracing such as availability of the decoder to an outsider. On the other hand, we stress that this work is motivated to design efficient black-box traitor tracing scheme with public/private tracing for specific FEs which are fully collusion resistant. We leave the problem of extending Zhandry's work to construct white-box traitor tracing for FE as an interesting future work.

2 Technical Overview

We start by reviewing the *embedded identity traitor tracing* (EI-TT) scheme of Goyal et al. [GKW19] which is referred to GKW-TT from now on. The tracing algorithm of GKW-TT is designed to trace traitors' identities directly in PKE via an intermediate primitive called *embedded identity private linear broadcast encryption* (EIPL-BE). We extend their framework from PKE to IPFE and define the notion of *embedded identity traceable IPFE* (EI-TIPFE) for tracing identities of traitors in an IPFE system.

We further generalize the notion of EI-TIPFE to *embedded identity traceable identity-based IPFE* (EI-TIBIPFE) which allows to trace users that belong to a specific group. We

generically construct EI-TIBIPFE from an intermediate primitive *embedded identity private linear IBIPFE* (EIPL-IBIPFE). This step is inspired from the approach of [GKW19]. We *emphasize* that EI-TIPFE is a particular case of EI-TIBIPFE and one can similarly achieve it from EIPL-IPFE (which is again a specific case of EIPL-IBIPFE).

2.1 The Tracing Mechanism of GKW-TT

Goyal et al. [GKW19] design EI-TT schemes from various standard assumptions. The core idea of [GKW19] was to extend the framework of *private linear broadcast encryption* (PL-BE) [BSW06] and introduce the notion of EIPL-BE. An EIPL-BE consists of a setup, encryption, special encryption, and decryption algorithms. The difference between PL-BE and EIPL-BE is in the special encryption which is associated with the index-position-bit tuple (i, ℓ, b), whereas in [BSW06] the special encryption is associated only with the index i. More formally, EIPL-BE works as follows:

Setup$(1^\lambda, n, 1^k) \to$ (msk, mpk, key): The setup algorithm outputs a master key pair and a special encryption key.

KeyGen(msk, j, id) $\to$ sk: It generates a secret key sk for the tuple (j, id).

Enc(mpk, M) $\to$ ct: The normal encryption algorithm encrypts a message M and outputs a ciphertext ct.

SplEnc(key, $M, (i, \ell, b)$) $\to$ ct: The special encryption algorithm encrypts a message M to index-position-bit tuple (i, ℓ, b).

Dec(sk, ct) $\to \zeta/\perp$: If ct is a ciphertext of normal encryption then the decryption recovers the message and if ct is a special encryption ciphertext then sk can decrypt it as long as one of the conditions $(j > i)$ or $(j = i \wedge \ell = \perp)$ or $(j = i \wedge \mathsf{id}_\ell = 1 - b)$ holds (where id_ℓ denotes the ℓ-th bit of id).

At a high level, the role of EIPL-BE is similar to the role of PL-BE in the tracing mechanism of [BSW06]. Apart from restricting decryption by secret keys $\{\mathsf{sk}_j\}_{j \le i}$ (as in [BSW06]), the extended functionality of EIPL-BE is capable of preventing a secret key $\mathsf{sk}_{j,\mathsf{id}}$ to decrypt a ciphertext associated with the tuple $(j, \ell, \mathsf{id}_\ell)$. The former feature enables the tracing algorithm to find out the indices of traitors and the later discover the corresponding identities. We discuss the security notions required for tracing in the context IBIPFE.

2.2 Definition of Embedded Identity Traceable IBIPFE

We first discuss the algorithms of *traceable IPFE* (TIPFE) (which does not embed user identities in the secret keys) introduced by Do et al. [DPP20]. A TIPFE consists of five PPT algorithms (Setup, KeyGen, Enc Dec, Trace). The setup algorithm receives the number of users n, the length of vectors m, and outputs the master secret key msk, a tracing key key, and the master public key mpk. In the key generation phase, a user gets a secret key $\mathsf{sk}_{\boldsymbol{u}}$ associated with a user index i and a predicate vector $\boldsymbol{u} \in \mathbb{Z}^m$. The encryption algorithm outputs the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ for the message vector $\boldsymbol{v} \in \mathbb{Z}^m$. In decryption, the secret key holder computes $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ from the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ using her/his secret key $\mathsf{sk}_{\boldsymbol{u}}$. The tracing algorithm, given oracle access to a pirated decoder box $\mathscr{D}_{\boldsymbol{u}}$ associated with a vector $\boldsymbol{u}$, uses (the tracing key) key, a tracing parameter $\epsilon(\cdot)$, a pair of message vectors $(\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$, and extracts a set of indices $T^{\mathsf{index}} \subset [n]$ related to the traitors.

The main downside of the tracing algorithm is that indices in T^{index} are traced back to the actual identities through a central *map*, which becomes problematic for many applications. To overcome this limitation, we extend the notion of TIPFE into *embedded identity TIPFE* (EI-TIPFE) where the users' secret keys are associated with index-identity pair (j, id) such that id can be a binary string of length k. Second and the essential change is in the tracing algorithm which now directly extracts a set T^{id} containing the identities of traitors. Hence, there is no need of such unnecessary *map* of TIPFE.

We note that the encryption of EI-TIPFE is performed independently of the sender's identity, hence receivers are unaware of the source of plaintexts. Moreover, in real applications, it is often the case that a group of users (e.g. employees) possesses a group identity (e.g. the company in which they work). This motivates us to define the notion of *embedded identity traceable IBIPFE* (EI-TIBIPFE) where the secret keys of users are additionally associated with a group identity $\mathsf{gid} \in \{0,1\}^{k'}$ and the ciphertexts are computed under a group identity gid'. The decryption successfully recovers the inner product if these two group identities are the same, i.e. $\mathsf{gid} = \mathsf{gid}'$. The tracing becomes more efficient since the pirated decoder box $\mathscr{D}_{\boldsymbol{u}}$ works with a specific gid and one need to only trace over the set of users that linked with the gid, instead of the set of all users in the system. More formally, our EI-TIBIPFE scheme is defined as follows:

Setup$(1^{\lambda}, n, 1^{k}, 1^{k'}, 1^{m}) \rightarrow (\mathsf{msk}, \mathsf{mpk}, \mathsf{key})$: The setup algorithm generates master key pairs and a tracing key.

KeyGen$(\mathsf{msk}, i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u}) \rightarrow \mathsf{sk}_{\boldsymbol{u}}$: It generates secret keys of users with index-identity pair (i, id) and a group identity gid.

Enc$(\mathsf{mpk}, \mathsf{gid}', \boldsymbol{v}) \rightarrow \mathsf{ct}_{\boldsymbol{v}}$: It encrypts a vector $\boldsymbol{v}$ under a group identity gid'.

Dec$(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) \rightarrow \zeta/\perp$: The decryption recovers the inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ if $\mathsf{gid} = \mathsf{gid}'$; otherwise returns $\perp$.

Trace$^{\mathscr{D}_{\boldsymbol{u}}}(\mathsf{key}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{gid}, \boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \rightarrow T^{\mathsf{id}}$: It outputs a set $T^{\mathsf{id}} \subset \{0,1\}^{k}$ of the identities of traitors belong to the group identity gid.

We say that the tracing is correct if it does not falsely accuse an honest user as traitor and T^{id} is a subset of the identities of released secret keys. EI-TIBIPFE can be viewed as a particular case of EI-TIBIPFE when $\mathsf{gid} = \mathsf{gid}'$ always holds. In other words, one can independently define EI-TIPFE from EI-TIBIPFE where KeyGen, Enc, Trace$^{\mathscr{D}_{\boldsymbol{u}}}$ do not take a group identity as input and decryption with honestly generated secret keys is always successful.

2.3 The framework of EIPL-IBIPFE for tracing identities in IBIPFE

The backbone of our EI-TIBIPFE construction is the notion of EIPL-IBIPFE. The concept of EIPL-IBIPFE is inspired by the primitive of EIPL-BE introduced by [GKW19] for tracing traitors in PKE. We extend their framework from PKE to FE, more specifically to IPFE or even richer functionality of IBIPFE. Instead of encrypting an integer message, EIPL-IBIPFE encrypts an integer vector from $\mathbb{Z}^m$ under a group identity gid' and generates secret keys corresponding to a vector of the same length for users with index-identity pair (j, id) belonging to a group having identity gid. The encryption has two modes – the *normal* mode is similar to the encryption of EI-TIBIPFE and the *special* mode is used for tracing purpose. More formally, our EIPL-IBIPFE is defined as follows:

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m) \to$ **(msk, mpk, key)**: The setup algorithm outputs a master key pair and a special encryption key.

KeyGen$(\mathsf{msk}, j, \mathsf{id}, \mathsf{gid}, \boldsymbol{u}) \to \mathsf{sk}_{\boldsymbol{u}}$: It generates the secret key $\mathsf{sk}_{\boldsymbol{u}}$ associated with the tuple (j, id) and a group identity gid.

Enc$(\mathsf{mpk}, \mathsf{gid}', \boldsymbol{v}) \to \mathsf{ct}_{\boldsymbol{v}}$: The normal encryption algorithm encrypts a message vector $\boldsymbol{v}$ under the group identity gid'.

SplEnc$(\mathsf{key}, \mathsf{gid}', \boldsymbol{v}, (i, \ell, b)) \to \mathsf{ct}_{\boldsymbol{v}}$: The special encryption algorithm encrypts a message vector $\boldsymbol{v}$ for index-position-bit tuple (i, ℓ, b). If (the special encryption key) key is publicly available then it is called *public* EIPL-IBIPFE; otherwise it is known as *private* EIPL-IBIPFE.

Dec$(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) \to \zeta/\perp$: If $\mathsf{ct}_{\boldsymbol{v}}$ is a ciphertext of normal encryption then the decryption recovers the message when $\mathsf{gid} = \mathsf{gid}'$ holds. On the other hand, the user can decrypt a special encryption ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ if $\mathsf{sk}_{\boldsymbol{u}}$ satisfies the conditions $(j > i)$ or $(j = i \wedge \ell = \perp)$ or $(j = i \wedge \mathsf{id}_\ell = 1 - b)$ where id_ℓ be the ℓ-th bit of id.

We now observe that EI-TIBIPFE can be directly obtained from EIPL-IBIPFE. The Setup, KeyGen, Enc, Dec algorithms of both the primitives works exactly in similar fashion. We briefly discuss the tracing procedure. Before that, we formalize the following security properties for our EIPL-IBIPFE by combining the security notions of two primitives EIPL-BE [GKW19] and IBIPFE [ACGU20] :

- **Normal-hiding.** $\mathsf{Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}) \approx_c \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (1, \perp, 0))$.
- **Index-hiding.** $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \perp, 0)) \approx_c \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*+1, \perp, 0))$ if an adversary is not given a key for $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$.
- **Lower identity-hiding.** $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \perp, 0)) \approx_c \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*))$ if an adversary is not given a key for $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = b^*$.
- **Upper identity-hiding.** $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*)) \approx_c \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*+1, \perp, 0))$ if an adversary is not given a key for $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = 1 - b^*$.
- **Message-hiding.** $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, (i^*, \perp, 0)) \approx_c \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(1)}, (i^*, \perp, 0))$ if all the secret keys associated to $(i \geq i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ satisfy the condition $\langle \boldsymbol{v}^{(0)}, \boldsymbol{u} \rangle = \langle \boldsymbol{v}^{(1)}, \boldsymbol{u} \rangle$.

We call an EIPL-IBIPFE selectively/adaptively secure subject to the selection of the challenge tuple $(\mathsf{gid}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$ by an adversary before/after the setup and pre-ciphertext key queries. The above security properties of EIPL-IBIPFE facilitates revealing a traitor's identity in a bit-by-bit manner. The role of a special encryption algorithm is similar to that of the indexed-encryption algorithm of [BSW06] except it provides an additional feature that disables the decryption ability of users upon a single bit of the identity. In more detail, the tracing mechanism follows a two-step process:

1. **Index tracing.** The first step is similar to the usual PI-BE or EIPL-BE [BSW06, GKW19], where the indices of dishonest users' are traced. Formally, for each indices $i \in [n+1]$, it finds the probability $\hat{p}_i^{\mathsf{ind}}$ of the decoder box for successfully decrypting special encryptions to the tuple $(i, \perp, 0)$. It outputs $\mathsf{Index} = \{i \in [n+1] : \hat{p}_i^{\mathsf{ind}}$ and $\hat{p}_{i+1}^{\mathsf{ind}}$ are noticeably far$\}$.
2. **Identity tracing.** The second step is a sub-search technique which is performed to trace the identity for each $i \in \mathsf{Index}$. It checks whether the ℓ-th bit in a (possibly) traitor's identity is zero or one for all index positions $\ell \in [k]$. Formally, for each $i \in \mathsf{Index}$, and $\ell \in [k]$, it finds the probability $\hat{q}_{i,\ell}^{\mathsf{id}}$ of the decoder box for successfully decrypting special encryptions to the tuple $(i, \ell, 0)$.

Finally, for each $\ell \in [k]$, it sets $\mathsf{id}_\ell = 0$ if $\hat{p}_i^{\mathsf{ind}}$ and $\hat{q}_{i,\ell}^{\mathsf{id}}$ are noticeably far; otherwise 1. The index tracing phase identifies the possible indices of traitors using the index-hiding property. In the second step, the lower identity-hiding and the upper identity-hiding properties ensure that estimate $\hat{q}_{i,\ell}^{\mathsf{id}}$ is either close to $\hat{p}_i^{\mathsf{ind}}$ or $\hat{p}_{i+1}^{\mathsf{ind}}$, which indeed enables it to extract the correct bit of id_ℓ with high probability.

We show that the tracing algorithm of our EI-TIBIPFE proceeds in a similar fashion as it is run in [GKW19], However, we need to be extra careful due to the extended functionality of IBIPFE. For instance, the "Index tracing" step requires to employ the message-hiding property that allows the decoder to decrypt a ciphertext in our setting. On the other hand, [GKW19] uses the message-hiding property to entirely restrict the decoder in decrypting a ciphertext.

Finally, one can observe that EIPL-IPFE is a particular case of EIPL-IBIPFE with $\mathsf{gid} = \mathsf{gid}'$, otherwise one may ignore the group identity from the above definition to translate it into EIPL-IPFE.

2.4 EIPL-IBIPFE from IBIPFE

We first describe our generic construction of EIPL-IBIPFE from any existing IBIPFE such as pairing or lattice-based IBIPFE of Abdalla et al. [ACGU20]. In an IBIPFE, secret keys are generated for an identity-vector pair $(\mathsf{gid}, \boldsymbol{u})$ and ciphertexts are computed for another identity-vector pair $(\mathsf{gid}', \boldsymbol{v})$ such that decryption recovers the inner product $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ if the two identities match, i.e. $\mathsf{gid} = \mathsf{gid}'$. Given a set of secret keys for $\{(\mathsf{gid}, \boldsymbol{u})\}$, it is hard to distinguish between encryptions of $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ under gid' if $\langle \boldsymbol{u}, \boldsymbol{v}_0 \rangle = \langle \boldsymbol{u}, \boldsymbol{v}_1 \rangle$ holds for all the secret keys whenever $\mathsf{gid} = \mathsf{gid}'$.

Our EIPL-IBIPFE construction is inspired by the approach of building EIPL-BE using only a PKE scheme [GKW19]. We replace the underlying PKE with an IBIPFE scheme, which transforms their generic EIPL-BE into an EIPL-IBIPFE. This construction provides useful insights for designing more efficient EIPL-IBIPFE from pairing (presented later in this section).

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m)$: $\{\mathsf{IBIPFE.msk}_{i,\ell,b}, \mathsf{IBIPFE.mpk}_{i,\ell,b}\} \leftarrow \mathsf{IBIPFE.Setup}\,(1^\lambda, 1^m, 1^{k'})$, Sets $\mathsf{mpk} = \{\mathsf{IBIPFE.mpk}_{i,\ell,b}\}_{(i,\ell,b)\in[n]\times[k]\times\{0,1\}}$, $\mathsf{key} = \mathsf{mpk}$ and $\mathsf{msk} = \{\mathsf{IBIPFE.msk}_{i,\ell,b}\}_{(i,\ell,b)\in[n]\times[k]\times\{0,1\}}$.

KeyGen$(\mathsf{msk}, i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$: $\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell} \leftarrow \mathsf{IBIPFE.KeyGen}\,(\mathsf{IBIPFE.msk}_{i,\ell,\mathsf{id}_\ell}, \mathsf{gid}, \boldsymbol{u})$ for all $\ell \in [k]$, $\mathsf{sk}_{\boldsymbol{u}} = \{\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell}\}_{\ell\in[k]}$.

Enc$(\mathsf{mpk}, \mathsf{gid}', \boldsymbol{v})$: $\{\boldsymbol{v}_{i,\ell}\}_{(i,\ell)\in([n]\times[k-1])} \leftarrow \mathbb{Z}^m$, $\boldsymbol{v}_{i,k} = \boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell}$, $\mathsf{IBIPFE.ct}_{i,\ell,b} \leftarrow \mathsf{IBIPFE.Enc}$ $(\mathsf{IBIPFE.mpk}_{i,\ell,b}, \mathsf{gid}', \boldsymbol{v}_{i,\ell})$, $\mathsf{ct}_{\boldsymbol{v}} = \{\mathsf{IBIPFE.ct}_{i,\ell,b}\}_{(i,\ell,b)\in([n]\times[k]\times\{0,1\})}$.

SplEnc$(\mathsf{key}, \mathsf{gid}', \boldsymbol{v}, (i^*, \ell^*, b^*))$: For $i \geq i^*, \boldsymbol{v}_{i,k} = \boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell}$ else $\boldsymbol{v}_{i,k} \leftarrow \mathbb{Z}^m$. If $(i,\ell,b) \neq (i^*, \ell^*, b^*)$, $\widetilde{\boldsymbol{v}}_{i,\ell,b} = \boldsymbol{v}_{i,\ell}$, else $\widetilde{\boldsymbol{v}}_{i,\ell,b} \leftarrow \mathbb{Z}^m$, $\mathsf{ct}_{i,\ell,b} \leftarrow \mathsf{IBIPFE.Enc}$ $(\mathsf{IBIPFE.mpk}_{i,\ell,b}, \mathsf{gid}', \widetilde{\boldsymbol{v}}_{i,\ell,b})$, $\mathsf{ct}_{\boldsymbol{v}} = \{\mathsf{ct}_{i,\ell,b}\}_{(i,\ell,b)\in([n]\times[k]\times\{0,1\})}$.

Dec$(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}})$: $\{\langle \boldsymbol{u}, \boldsymbol{v}_{i,\ell} \rangle\}_{\ell\in[k]} \leftarrow \mathsf{IBIPFE.Dec}(\{\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell}\}_{\ell\in[k]}, \{\mathsf{IBIPFE.ct}_{i,\ell,\mathsf{id}_\ell}\}_{\ell\in[k]})$, if $\mathsf{gid} = \mathsf{gid}'$ and $\forall\, i \in [n]$, outputs $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{\ell\in[k]} \langle \boldsymbol{u}, \boldsymbol{v}_{i,\ell} \rangle$.

We note that it is sufficient to generate $2nk$ many key pairs of IBIPFE since the group identity (having size k' in the construction) is not required to trace. Thus, the size of the ciphertexts in our EIPL-IBIPFE grows linearly with the maximum number of users n and the length of the user identities k. In other words, each n slot of the ciphertext contains k independent and disjoint components of the IBIPFE sub-ciphertexts. Thus, a user can perform

decryption for each n slot by only looking at its dedicated IBIPFE component in the ciphertext. Since all the IBIPFE ciphertexts are independently created, the security of EIPL-IBIPFE directly follows from the security of IBIPFE. For instance, consider the message-hiding security where special encryptions of two message vectors $\boldsymbol{v}^{(0)}$ and $\boldsymbol{v}^{(1)}$ to the tuple $(i^*, \perp, 0)$ under gid' must be indistinguishable if all secret key queries for the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ satisfies the condition that whenever $i \geq i^*$ and $\mathsf{gid} = \mathsf{gid}^*$, it holds that $\langle \boldsymbol{v}^{(0)}, \boldsymbol{u} \rangle = \langle \boldsymbol{v}^{(1)}, \boldsymbol{u} \rangle$. Note that, for $i \geq i^*$, the vector $\boldsymbol{v}^{(\beta)}_{i,k} = \boldsymbol{v}^{(\beta)} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell}$ is encrypted under $\mathsf{mpk}_{i,k,b}$ using IBIPFE where β is either 0 or 1. Therefore, for all $i \geq i^*$, it holds that $\langle \boldsymbol{v}^{(0)}_{i,k}, \boldsymbol{u} \rangle = \langle \boldsymbol{v}^{(1)}_{i,k}, \boldsymbol{u} \rangle$, and hence the message-hiding security follows from the security of IBIPFE.

2.5 EIPL-IPFE from pairing

The generic construction discussed above is not a desirable solution to obtain EI-TIBIPFE as the ciphertext size linearly grows with the number of users in the system. In search of a more efficient solution, we investigate non-generic group based constructions of EIPL-IBIPFE. However, first we consider a simpler situation where the group identity is absent. That is, we present a pairing-based construction of EIPL-IPFE which will lead to an EI-TIPFE scheme.

Our starting point is the pairing-based EIPL-BE scheme by Goyal et al. [GKW19], which builds upon the PL-BE scheme by Boneh et al. [BSW06]. Let us first recall the EIPL-BE scheme. In what follows, we denote an element g of the group $\mathbb{G}$ of order $N = p \cdot q$ (where p, q are the primes) and a vector $\boldsymbol{a} = (a_1, \ldots, a_m)$, we denote $(g^{a_1}, \ldots, g^{a_m})$ by $g^{\boldsymbol{a}}$. For two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$, we denote $\langle g^{\boldsymbol{a}}, \boldsymbol{b} \rangle = g^{\langle \boldsymbol{a}, \boldsymbol{b} \rangle}$. Let e be a bilinear map define as $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, and $\mathbb{G}_p, \mathbb{G}_q$ be the subgroups of $\mathbb{G}$ of orders p and q respectively. Suppose, there are n parties indexing each by $i \in [n]$ which is represented as a pair $(x, y) \in [\sqrt{n}] \times [\sqrt{n}]$. We say that $i_1 \equiv (x_1, y_1) > i_2 \equiv (x_2, y_2)$ if either $x_1 > x_2$ or $(x_1 = x_2 \wedge y_1 > y_2)$.

Setup($1^\lambda, n, 1^k$): Chooses $g_p, h_p \leftarrow \mathbb{G}_p; g_q, h_q \leftarrow \mathbb{G}_q : g = g_p g_q, h = h_p h_q$; $\beta \leftarrow \mathbb{Z}_N$ and for all $x \in [\sqrt{n}], r_x, \alpha_x \leftarrow \mathbb{Z}_N$. Compute $E_q = g_q^{\beta}, E_x = g^{r_x}, F_x = h^{r_x}, G_x = e(g,g)^{\alpha_x}, E_{q,x} = g_q^{\beta r_x}, F_{q,x} = h_q^{\beta r_x}, G_{q,x} = e(g_q, g_q)^{\beta \alpha_x}$; For each $(y, \ell, b) \in ([\sqrt{n}] \times [k] \times \{0,1\}), c_{y,\ell,b}, \delta_{\ell,b} \leftarrow \mathbb{Z}_N$, $\gamma_{\ell,b} \leftarrow \mathbb{Z}_p$. $H_{y,\ell,b} = g^{c_{y,\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}}, \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}$. Outputs $\mathsf{key} = \mathsf{mpk}$,

$$\mathsf{mpk} = \left(E_q, \left\{ \begin{matrix} E_x, & F_x, & G_x, \\ E_{q,x}, & F_{q,x}, & G_{q,x} \end{matrix} \right\}_x, \left\{ \begin{matrix} H_{y,\ell,b}, V_{\ell,b}, \\ \widetilde{V}_{\ell,b} \end{matrix} \right\}_{y,\ell,b} \right); \mathsf{msk} = \left(\begin{matrix} \{\alpha_x, r_x\}_x, \\ \{c_{y,\ell,b}\}_{y,\ell,b} \end{matrix} \right)$$

KeyGen(msk, $\boldsymbol{i}$, id): $i \equiv (x, y)$, $K = g^{\alpha_x + r_x \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}}$. Outputs $\mathsf{sk} = (x, y, \mathsf{id}, K)$.

Enc(mpk, $\boldsymbol{M}$): $\mathsf{ct} \leftarrow \mathsf{SplEnc}(\mathsf{key}, M, (1, \perp, 0))$. Outputs ct_M.

SplEnc(key, $\boldsymbol{M} \in \mathbb{G}_T, (\boldsymbol{i}^*, \boldsymbol{\ell}^*, \boldsymbol{b}^*)$): $i^* \equiv (x^*, y^*)$; $\tau, t \leftarrow \mathbb{Z}_N$; for all $x \in [\sqrt{n}], s_x, e_x, f_x \leftarrow \mathbb{Z}_N$; $w_{y,\ell,b}, v_{y,\ell,b} \leftarrow \mathbb{Z}_N \; \forall (y, \ell, b) \in [\sqrt{n}] \times [k] \times \{0,1\}$.

	R_x	$\widetilde{R}_x$	A_x	I_x	role in decryption
$x > x^*$	$E_{q,x}^{s_x}$	$F_{q,x}^{s_x\tau}$	$E_q^{s_x t}$	$M \cdot G_{q,x}^{ts_x}$	successful decryption for $i > i^*$
$x = x^*$	$E_x^{s_x}$	$F_x^{s_x\tau}$	$g^{s_x t}$	$M \cdot G_x^{ts_x}$	successful decryption for $(i > i^*) \vee ((i,\ell) = (i^*, \perp)) \vee ((i, \mathsf{id}_\ell) = (i^*, 1-b^*))$
$x < x^*$	g^{s_x}	$h^{s_x\tau}$	$g^{e_x t}$	$e(g,g)^{f_x}$	unsuccessful decryption
$(y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*))$			$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$	successful decryption for $(x > x^*) \vee ((i,\ell) = (i^*, \perp)) \vee ((i, \mathsf{id}_\ell) = (i^*, 1-b^*))$
			$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$	
$(y < y^*) \vee ((y,\ell,b) = (y^*, \ell^*, b^*))$			$H_{y,\ell,b}^t \cdot h^{\tau w_{y,\ell,b}} \cdot V_{\ell,b}^{\tau v_{y,\ell,b}}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$	successful decryption for $x > x^*$

Outputs $\mathsf{ct} = \left(\{R_x, \widetilde{R}_x, A_x, I_x\}_x, \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{y,\ell,b}\right)$

Dec(sk, ct): $M = \frac{I_x \cdot e(R_x, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell})}{e(\widetilde{R}_x, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}) \cdot e(K, A_x)}$

The EIPL-BE of [GKW19] makes use of the generic construction of EIPL-BE from PKE and consider $2k$ different subsystems of the PL-BE scheme by Boneh et al. [BSW06]. Note that, while all the subsystems share the same $\{\alpha_x, r_x\}_{x \in [\sqrt{n}]}$, each of it possesses own $\{c_y = \sum_\ell c_{y,\ell,b}\}$ values so that the key generation algorithm can select appropriate $\{c_y\}$ values for each identity id. This prevents mixing terms of different secret keys to create a hybrid key. Inspired from the approach of EIPL-BE, we upgrade the system to make it capable of encrypting vectors instead of a single integer. To do so, we make use of the DDH-based IPFE scheme by Agrawal et al. [ALS16]. More specifically, the components $E_x, F_x, G_x, E_{q,x}, F_{q,x}, G_{q,x}$ along with α_x, r_x are now upgraded to vectors. Additionally, we consider a vector $\boldsymbol{\psi}_x \leftarrow \mathbb{Z}_N^m$ (whose role is similar to α_x in the GKW scheme). It helps us to argue the adaptive security of the *implicit* IPFE scheme. Armed with these ideas, we construct EIPL-IPFE as follows:

Setup($1^\lambda, n, 1^k, 1^m$): Samples $g_p, h_p, f_p \leftarrow \mathbb{G}_p; g_q, h_q, f_q \leftarrow \mathbb{G}_q : g = g_p g_q, h = h_p h_q, f = f_p f_q$; For all $x \in [\sqrt{n}]$, chooses $\beta \leftarrow \mathbb{Z}_N \boldsymbol{r}_x, \boldsymbol{\alpha}_x, \boldsymbol{\psi}_x \leftarrow \mathbb{Z}_N^m$, computes $E_q = g_q^\beta, Z_q = f_q^\beta, \boldsymbol{E}_x = g^{\boldsymbol{r}_x}, \boldsymbol{F}_x = h^{\boldsymbol{r}_x}, \boldsymbol{G}_x = e(g,g)^{\boldsymbol{\alpha}_x}, \boldsymbol{W}_x = e(f,f)^{\boldsymbol{\psi}_x}, \boldsymbol{E}_{q,x} = g_q^{\beta \boldsymbol{r}_x}, \boldsymbol{F}_{q,x} = h_q^{\beta \boldsymbol{r}_x}, \boldsymbol{G}_{q,x} = e(g_q, g_q)^{\beta \boldsymbol{\alpha}_x}, \boldsymbol{W}_{q,x} = e(f_q, f_q)^{\beta \boldsymbol{\psi}_x}$; For all $(y, \ell, b) \in [\sqrt{n}] \times [k] \times \{0,1\}, \delta_{\ell,b}, c_{y,\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p$, computes $H_{y,\ell,b} = g^{c_{y,\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}}, \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}$. Sets key = mpk,

$$\mathsf{mpk} = \left(\left\{ \begin{matrix} g, h, f, \\ E_q, Z_q, \end{matrix} \right\}, \left\{ \begin{matrix} \boldsymbol{E}_x, & \boldsymbol{F}_x, & \boldsymbol{G}_x, & \boldsymbol{W}_x \\ \boldsymbol{E}_{q,x}, & \boldsymbol{F}_{q,x}, & \boldsymbol{G}_{q,x}, & \boldsymbol{W}_{q,x} \end{matrix} \right\}_x, \left\{ \begin{matrix} H_{y,\ell,b}, \\ \widetilde{V}_{\ell,b}, V_{\ell,b} \end{matrix} \right\}_{y,\ell,b} \right), \mathsf{msk} = \left(\begin{matrix} \{\boldsymbol{\alpha}_x, \boldsymbol{r}_x, \boldsymbol{\psi}_x\}_x \\ \{c_{y,\ell,b}\}_{y,\ell,b} \end{matrix} \right)$$

KeyGen(msk, $\boldsymbol{i}$, id, $\boldsymbol{u}$): $i \equiv (x,y)$, $K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle + \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}}; K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$. Outputs $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, K_1, K_2)$.

Enc(mpk, $\boldsymbol{v}$): $\mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \boldsymbol{v}, (1, \perp, 0))$. Outputs $\mathsf{ct}_{\boldsymbol{v}}$.

SplEnc(key, $\boldsymbol{v}$, $(\boldsymbol{i}^*, \boldsymbol{\ell}^*, \boldsymbol{b}^*)$): $i^* \equiv (x^*, y^*)$, $\tau, t \leftarrow \mathbb{Z}_N$; for all $x \in [\sqrt{n}]$, samples $s_x, \kappa_x, e_x, d_x \leftarrow \mathbb{Z}_N; \boldsymbol{\sigma}_x, \boldsymbol{f}_x \leftarrow \mathbb{Z}_N^m; w_{y,\ell,b}, v_{y,\ell,b} \leftarrow \mathbb{Z}_N$ for all $(y, \ell, b) \in [\sqrt{n}] \times [k] \times \{0,1\}$.

	$\boldsymbol{R}_x$	$\widetilde{\boldsymbol{R}}_x$	A_x	B_x	$\boldsymbol{I}_x$	role in decryption
$x > x^*$	$\boldsymbol{E}_{q,x}^{s_x}$	$\boldsymbol{F}_{q,x}^{s_x\tau}$	$E_q^{s_x t}$	$Z_q^{\kappa_x t}$	$e(g_q,g_q)^{\boldsymbol{v}} \cdot \boldsymbol{G}_{q,x}^{t s_x} \cdot \boldsymbol{W}_{q,x}^{t\kappa_x}$	successful decryption for $i > i^*$
$x = x^*$	$\boldsymbol{E}_x^{s_x}$	$\boldsymbol{F}_x^{s_x\tau}$	$g^{s_x t}$	$f^{\kappa_x t}$	$e(g,g)^{\boldsymbol{v}} \cdot \boldsymbol{G}_x^{t s_x} \cdot \boldsymbol{W}_x^{t\kappa_x}$	successful decryption for $(i > i^*) \vee ((i,\ell) = (i^*,\perp)) \vee ((i,\mathsf{id}_\ell) = (i^*, 1-b^*))$
$x < x^*$	$g^{\boldsymbol{\sigma}_x}$	$h^{\boldsymbol{\sigma}_x\tau}$	$g^{e_x t}$	f^{d_x}	$e(g,g)^{f_x} \cdot e(f,f)^{f_x}$	unsuccessful decryption

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$	
$(y > y^*) \vee (y = y^* \wedge (\ell,b) \neq (\ell^*,b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$	successful decryption for $(x > x^*) \vee ((i,\ell) = (i^*,\perp)) \vee ((i,\mathsf{id}_\ell) = (i^*, 1-b^*))$
$(y < y^*) \vee ((y,\ell,b) = (y^*,\ell^*,b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau} \cdot V_{\ell,b}^{\tau v_{y,\ell,b}}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$	successful decryption for $x > x^*$

Outputs $\mathsf{ct}_{\boldsymbol{v}} = \left(\{\boldsymbol{R}_x, \widetilde{\boldsymbol{R}}_x, A_x, B_x, \boldsymbol{I}_x\}_x, \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{y,\ell,b}\right)$

Dec($\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}$): $\eta = \dfrac{\langle \boldsymbol{I}_x, \boldsymbol{u}\rangle \cdot e(\boldsymbol{R}_x, \prod_{\ell\in[k]} C_{y,\ell,\mathsf{id}_\ell}^{\boldsymbol{u}})}{e(\widetilde{\boldsymbol{R}}_x, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{\boldsymbol{u}}) \cdot e(K_1, A_x) \cdot e(K_2, B_x)}$. Outputs $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \log \eta$

The normal hiding security directly follows from the construction of the EIPL-IPFE scheme. The main intuition behind the index-hiding security proof is that if an adversary does not have a secret key for the index $i^* = (x^*, y^*)$ then the factor of $C_{y^*,\ell,b}$ which belongs to $\mathbb{G}_p$ can be chosen undetectably, added, and removed. Similar techniques are used while proving the lower identity-hiding and upper identity-hiding security notions. In the message-hiding security, an adversary can not distinguish between the special encryption of the challenge vectors $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ to the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$ with the restriction that the adversary is allowed to query secret keys for $i \geq i^*$ satisfying $\langle \boldsymbol{v}^{(0)}, \boldsymbol{u}\rangle = \langle \boldsymbol{v}^{(1)}, \boldsymbol{u}\rangle$. We encounter some challenges while plugging the DDH-based IPFE of [ALS16] into our settings and discuss why the plain DBDH assumption is not sufficient to prove the message-hiding security. The main difference is that there are several IPFE systems encrypting the message vector for each index-factor $x \in [\sqrt{n}]$ where some of these use $\mathbb{G}_q$ as the base group (when $x > x^*$) and others use the whole group $\mathbb{G}$ (when $x = x^*$). Furthermore, all of these systems share the same randomness t. Thus, we can not apply the plain DBDH assumption to argue the message-hiding security. Hence, we consider the joint-D3DH assumption inspired from the joint-SXDH assumption proposed by Lin et al. [LV16]. The joint-D3DH assumption helps us simulating terms like g_q^t, g^t with the same randomness $t \leftarrow \mathbb{Z}_N$, which in turn abets to overcome the barrier faced in the security reduction. Since for each $i \leq i^*$, it holds that $\langle \boldsymbol{v}^{(0)}, \boldsymbol{u}\rangle = \langle \boldsymbol{v}^{(1)}, \boldsymbol{u}\rangle$, we can argue the message-hiding security of EIPL-IPFE by leveraging the security of the *implicit* IPFE system for each index-factor $x \in [\sqrt{n}]$ for which decryption is successful. We provide a detail security analysis in Sec. 6.2. Finally, we note that the size of the ciphertext grows linearly with $\sqrt{n}$ and $\sqrt{k}$, similar to previous TT schemes [BSW06, GKW19]. Next, we discuss how to extend the EIPL-IPFE to EIPL-IBIPFE for adding the group identities into the system.

2.6 Extending EIPL-IPFE to EIPL-IBIPFE

The notion of IBIPFE provides more finer access control than IPFE and, hence it is more challenging to construct. A natural first attempt is to build EIPL-IBIPFE generically from EIPL-IPFE and an IBE scheme. However, any such attempt would fail to provide the message-hiding security due to common mix and match attacks. More specifically, a *mixed* secret key

obtained by combining an authorized IBE key with an unauthorized IPFE key can be used to decrypt an undesirable ciphertext. Thus, it is advisable to build such a scheme in a non-generic manner.

It is not hard to see that the above construction of EIPL-IPFE (or EIPL-BE) encrypts the message vectors in the target group. Thus, one needs to have an IBIPFE scheme based on a target group assumption so that it is well fitted into the GKW EIPL-BE system. However, all known pairing-based IBIPFE schemes [AGT21, ACGU20] are based on dual system encryption mechanisms and hence naturally depend on various source group assumptions (such as subgroup decision assumptions). We follow a two step approach – in the first step, we construct an IBIPFE scheme based on the plain DBDH assumption and then in the second step, we plug in our IBIPFE into the GKW EIPL-BE system – to achieve our goal of building EIPL-IBIPFE. Note that the second step indeed takes inspiration from the transformation of EIPL-IPFE from EIPL-BE.

Step I. IBIPFE from a target group assumption. Our starting point is the Water's IBE [Wat05] based on the plain DBDH assumption in the standard model. We upgrade their scheme in a natural way to enable encrypting vectors under a given identity.

Setup$(1^\lambda, 1^{k'}, 1^m)$: Samples $g, g_2, u' \leftarrow \mathbb{G}$, $\boldsymbol{u} = (u_i) \leftarrow \mathbb{G}^{k'}$, $\boldsymbol{\alpha} \leftarrow \mathbb{Z}_p^m$. $\boldsymbol{g}_1 = g^{\boldsymbol{\alpha}}, \boldsymbol{g}_2 = g_2^{\boldsymbol{\alpha}}$. Outputs $\mathsf{mpk} = (\boldsymbol{g}_1, g_2, u', \boldsymbol{u}, g), \mathsf{msk} = \boldsymbol{g}_2$.

KeyGen$(\mathsf{msk}, \mathsf{gid} \in \{0,1\}^{k'}, \boldsymbol{y})$: $\mathcal{V} = \{i \in [k'] : \mathsf{gid}_i = 1\}$, $\mathrm{H}(\mathsf{gid}) = u' \prod_{i \in \mathcal{V}} u_i \in \mathbb{G}$, $r \leftarrow \mathbb{Z}_p$. $d_1 = \langle \boldsymbol{g}_2, \boldsymbol{y} \rangle \cdot \mathrm{H}(\mathsf{gid})^r, d_2 = g^r$. Outputs $\mathsf{sk}_{\boldsymbol{y}} = (d_1, d_2)$.

Enc$(\mathsf{mpk}, \mathsf{gid}', \boldsymbol{x})$: Samples $t \leftarrow \mathbb{Z}_p$. $\boldsymbol{C}_1 = e(g, g_2)^{\boldsymbol{\alpha} t + \boldsymbol{x}}, C_2 = g^t, C_3 = \mathrm{H}(\mathsf{gid}')^t$. Outputs $\mathsf{ct}_{\boldsymbol{x}} = (\boldsymbol{C}_1, C_2, C_3)$.

Dec$(\mathsf{sk}_{\boldsymbol{y}}, \mathsf{ct}_{\boldsymbol{x}})$: If $\mathsf{gid} = \mathsf{gid}'$, then $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \langle \boldsymbol{C}_1, \boldsymbol{y} \rangle \cdot \frac{e(d_2, C_3)}{e(d_1, C_2)}$.

At a very high level, for proving the indistinguishability security of the above IBIPFE, we partially rely on the ideas of [AGT21] where they use dual system encryption techniques to prove the security of their attribute-based IPFE scheme based on source group assumptions. However, we aim to prove security based on the plain DBDH assumption, and hence there is no known way to use dual system encryption methodologies with such an assumption. We consider a sequence of hybrids and suppose that the challenge message vectors are $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}$ and the challenge identity is gid^*. The reduction begins by sampling a random *orthogonal* (full rank) matrix $\mathbf{F}$ satisfying the condition $\mathbf{F} \cdot (\boldsymbol{x}^{(0)} - \boldsymbol{x}^{(1)}) = \boldsymbol{e}_1$ where $\boldsymbol{e}_1$ is the first canonical basis vector. Accordingly the master key component $\boldsymbol{\alpha}$ is switched to $\mathbf{F}^\top \tilde{\boldsymbol{\alpha}}$. As the full rank matrix $\mathbf{F}$ is chosen uniformly at random, this transformation is statistically indistinguishable to the adversary's view. We observe that the challenge vectors are used in this hybrid to generate the master key pairs. Hence, we consider the selective security model where the adversary is restricted to submit the challenge messages before seeing any public parameter of the system. In the next hybrid, we use the DBDH assumption in order to hide the information of the challenge bit. Given a DBDH instance $(g^a, g^b, g^c, g_T^{abc})$, the adversary extends the group elements into vectors as $g^{\boldsymbol{a}}, g^{\boldsymbol{b}}, g^{\boldsymbol{c}}$ where $\boldsymbol{a} = (a, a_2, \ldots, a_m), \boldsymbol{b} = (b, \ldots, b), \boldsymbol{c} = (c, \ldots, c)$. It allows us to embed the DBDH-instance into the master keys. In the secret key query phase, we define identity encoding functions based on gid^* (similar to [Wat05]) to correctly simulate the accepting and non-accepting key queries. Finally, to simulate the challenge ciphertext we implicitly set $g_2 = g^b$ and $t = c$ so that ciphertext component $\boldsymbol{C}_1$ transforms

to $e(g,g)^{\mathbf{F}^\top\cdot(\boldsymbol{w}-\mathfrak{b}b\boldsymbol{e}_1+b\mathbf{F}\boldsymbol{x}^{(0)})}$ due to the choice of F, where $\boldsymbol{w}$ represents the component wise multiplication of the vectors $\boldsymbol{a},\boldsymbol{b}$ and $\boldsymbol{c}$. Now, observe that the challenge bit $\mathfrak{b}$ only occurs in the first entry of $(\boldsymbol{w}-\mathfrak{b}b\boldsymbol{e}_1+b\mathbf{F}\boldsymbol{x}^{(0)})$ and at the same time the DBDH-challenge element abc is encoded in the first entry of $\boldsymbol{w}$. Hence, the security of our IBIPFE follows from the plain DBDH assumption. In Appendix B, we give the full security analysis with selective identity for simplicity of exposition, however, we emphasize that using the techniques of [Wat05, BR09] one can prove the security with adaptive identity.

Step II. EIPL-IBIPFE from EIPL-BE and our IBIPFE. In the second step, we combine the GKW EIPL-BE and our IBIPFE in a similar fashion that has been followed while building the EIPL-IPFE. Recall that, in EIPL-IBIPFE, the users are linked with different group identities. Therefore, the secret key of a user is now associated with a user's identity id, a group identity gid′, and the message vector $\boldsymbol{v}$ is encrypted under a group identity gid′. The encoding H(gid′) appears the groups $\mathbb{G}$ and $\mathbb{G}_q$ during encryption. Thus, we can not directly use our IBIPFE for building EIPL-IBIPFE. To overcome this obstacle, we define a projection H_q of the identity encoding function H into the subgroup $\mathbb{G}_q$ of $\mathbb{G}$ as follows:

$$\text{Given group elements: } \vartheta'_p,\{\vartheta_{p,i}\}_{i\in[k']},\vartheta'_q,\{\vartheta_{q,i}\}_{i\in[k']} \text{ with } \vartheta'=\vartheta'_p\vartheta'_q,\vartheta_i=\vartheta_{p,i}\vartheta_{q,i}$$
$$\text{define: } \mathrm{H}(\mathsf{gid})=\vartheta'\prod_{i\in\mathcal{V}}\vartheta_i,\mathrm{H}_q(\mathsf{gid})=\vartheta'_q\prod_{i\in\mathcal{V}}\vartheta_{q,i}$$

where $\mathcal{V}=\{i\in[k']:\mathsf{gid}_i=1\}$. Equipped with this ideas, we describe below our EIPL-IBIPFE:

Setup$(1^\lambda,n,1^k,1^{k'},1^m)$: Samples $g_p,h_p,f_p,\vartheta'_p,\vartheta_{p,i}\leftarrow\mathbb{G}_p$ for all $i\in[k']$, $g_q,h_q,f_q,\vartheta'_q,\vartheta_{q,i}\leftarrow\mathbb{G}_q$ for all $i\in[k']$ such that $g=g_pg_q,h=h_ph_q,f=f_pf_q,\vartheta'=\vartheta'_p\vartheta'_q,\boldsymbol{\vartheta}=(\vartheta_{p,i}\vartheta_{q,i})_i$; For all $x\in[\sqrt{n}],\beta,\widehat{r}\leftarrow\mathbb{Z}_N,\boldsymbol{r}_x,\boldsymbol{\alpha}_x,\boldsymbol{\psi}_x\leftarrow\mathbb{Z}_N^m,E_q=g_q^\beta,\boldsymbol{E}_x=g^{\widehat{r}\boldsymbol{r}_x},\boldsymbol{F}_x=h^{\widehat{r}\boldsymbol{r}_x},\boldsymbol{G}_x=e(g,g)^{\boldsymbol{\alpha}_x},\boldsymbol{Y}_x=g^{\boldsymbol{\psi}_x},\boldsymbol{E}_{q,x}=g_q^{\beta\widehat{r}\boldsymbol{r}_x},\boldsymbol{F}_{q,x}=h_q^{\beta\widehat{r}\boldsymbol{r}_x},\boldsymbol{G}_{q,x}=e(g_q,g_q)^{\beta\boldsymbol{\alpha}_x},\boldsymbol{Y}_{q,x}=g_q^{\beta\boldsymbol{\psi}_x}$; For each $(y,\ell,b)\in[\sqrt{n}]\times[k]\times\{0,1\}$, samples $\delta_{\ell,b},c_{y,\ell,b}\leftarrow\mathbb{Z}_N,\gamma_{\ell,b}\leftarrow\mathbb{Z}_p$ and computes $H_{y,\ell,b}=g^{c_{y,\ell,b}},V_{\ell,b}=h^{\delta_{\ell,b}},\widetilde{V}_{\ell,b}=g^{\delta_{\ell,b}}g_p^{\gamma_{\ell,b}}$. Outputs key = mpk,

$$\mathsf{mpk}=\left(\begin{pmatrix}E_q,\vartheta',\vartheta_q'^{\beta},\\ \boldsymbol{\vartheta},\{\vartheta_{q,i}^{\beta}\}_i\end{pmatrix},\left\{\begin{matrix}\boldsymbol{E}_x,&\boldsymbol{F}_x,&\boldsymbol{G}_x,&\boldsymbol{Y}_x\\ \boldsymbol{E}_{q,x},&\boldsymbol{F}_{q,x},&\boldsymbol{G}_{q,x},&\boldsymbol{Y}_{q,x}\end{matrix}\right\}_x,\left\{\begin{matrix}H_{y,\ell,b}\\ V_{\ell,b},\widetilde{V}_{\ell,b}\end{matrix}\right\}_{y,\ell,b}\right),\ \mathsf{msk}=\begin{pmatrix}\{\boldsymbol{\alpha}_x,\boldsymbol{r}_x,\boldsymbol{\psi}_x\}_x,\\ \widehat{r},\{c_{y,\ell,b}\}_{y,\ell,b}\end{pmatrix}$$

KeyGen$(\mathsf{msk},\boldsymbol{i},\mathsf{id},\ \mathsf{gid},\boldsymbol{u})$: $i\equiv(x,y),\mathcal{V}=\{i\in[k']:\mathsf{gid}_i=1\}$, $\mathrm{H}(\mathsf{gid})=\vartheta'\prod_{i\in\mathcal{V}}\vartheta_i$; computes $K_1=g^{\langle\boldsymbol{\alpha}_x,\boldsymbol{u}\rangle+\widehat{r}\langle\boldsymbol{r}_x,\boldsymbol{u}\rangle\sum_{\ell\in[k]}c_{y,\ell,\mathsf{id}_\ell}};K_2=f^{\langle\boldsymbol{\psi}_x,\boldsymbol{u}\rangle}\cdot\mathrm{H}(\mathsf{gid})^{\widehat{r}},K_3=g^{\widehat{r}}$. Sets $\mathsf{sk}_{\boldsymbol{u}}=(x,y,\mathsf{id},\mathsf{gid},K_1,K_2,K_3)$.

Enc$(\mathsf{mpk},\mathsf{gid}',\boldsymbol{v})$: $\mathsf{ct}_{\boldsymbol{v}}\leftarrow\mathsf{SplEnc}(\mathsf{key},\boldsymbol{v},(1,\perp,0))$. Outputs $\mathsf{ct}_{\boldsymbol{v}}$.

SplEnc$(\mathsf{key},\mathsf{gid}',\boldsymbol{v},(\boldsymbol{i}^*,\boldsymbol{\ell}^*,\boldsymbol{b}^*))$: $i^*\equiv(x^*,y^*),\tau,t\leftarrow\mathbb{Z}_N$. For each $x\in[\sqrt{n}]$, samples $s_x,d_x,e_x\leftarrow\mathbb{Z}_N;\boldsymbol{\sigma}_x,\boldsymbol{f}_x\leftarrow\mathbb{Z}_N^m$, randomly chooses $w_{y,\ell,b},v_{y,\ell,b}\leftarrow\mathbb{Z}_N$ for all $(y,\ell,b)\in[\sqrt{n}]\times[k]\times\{0,1\}$. Returns $\mathsf{ct}_{\boldsymbol{v}}=\left(\{\boldsymbol{R}_x,\widetilde{\boldsymbol{R}}_x,A_x,B_x,\boldsymbol{I}_x\}_x,\{C_{y,\ell,b},\widetilde{C}_{y,\ell,b}\}_{y,\ell,b}\right)$

Dec$(\mathsf{sk}_{\boldsymbol{u}},\mathsf{ct}_{\boldsymbol{v}})$: If $\mathsf{sk}_{\boldsymbol{u}}$ is authorized, then it computes

$$\eta=\frac{\langle\boldsymbol{I}_x,\boldsymbol{u}\rangle\cdot e(\boldsymbol{R}_x,\prod_{\ell\in[k]}C_{y,\ell,\mathsf{id}_\ell}^{\boldsymbol{u}})}{e(\widetilde{\boldsymbol{R}}_x,\prod_{\ell\in[k]}\widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{\boldsymbol{u}})\cdot e(K_1,A_x)\cdot e(K_2,A_x)\cdot e(K_3,B_x)},\text{outputs }\langle\boldsymbol{u},\boldsymbol{v}\rangle=\log\eta$$

We consider selective security for our EIPL-IBIPFE where the adversary submits both the challenge message vectors and the challenge group identity before receiving any public parameter of the system. The security analysis is more involved and challenging in EIPL-IBIPFE.

	$\boldsymbol{R}_x$	$\widetilde{\boldsymbol{R}}_x$	A_x	B_x	$\boldsymbol{I}_x$	role in decryption
$x > x^*$	$\boldsymbol{E}_{q,x}^{s_x}$	$\boldsymbol{F}_{q,x}^{s_x\tau}$	$E_q^{s_x t}$	$\mathrm{H}_q(\mathsf{gid}')^{\beta s_x t}$	$e(g_q, g_q)^{\boldsymbol{v}} \cdot \boldsymbol{G}_{q,x}^{t s_x} \cdot e(f_q, \boldsymbol{Y}_{q,x})^{t s_x}$	successful decryption for $(i > i^*)$ with $\mathsf{gid} = \mathsf{gid}'$
$x = x^*$	$\boldsymbol{E}_x^{s_x}$	$\boldsymbol{F}_x^{s_x\tau}$	$g^{s_x t}$	$\mathrm{H}(\mathsf{gid}')^{s_x t}$	$e(g,g)^{\boldsymbol{v}} \cdot \boldsymbol{G}_x^{t s_x} \cdot e(f, \boldsymbol{Y}_x)^{t s_x}$	successful decryption for $(i > i^*) \vee ((i,\ell) = (i^*, \perp)) \vee ((i, \mathsf{id}_\ell) = (i^*, 1-b^*))$ with $\mathsf{gid} = \mathsf{gid}'$
$x < x^*$	$g^{\boldsymbol{\sigma}_x}$	$h^{\boldsymbol{\sigma}_x\tau}$	$g^{e_x t}$	$\mathrm{H}(\mathsf{gid}')^{d_x}$	$e(g,g)^{\boldsymbol{f}_x} \cdot e(f,f)^{\boldsymbol{f}_x}$	unsuccessful decryption

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$	
$(y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$	successful decryption $(x > x^*) \vee ((i,\ell) = (i^*, \perp)) \vee ((i, \mathsf{id}_\ell) = (i^*, 1-b^*))$
$(y < y^*) \vee ((y,\ell,b) = (y^*, \ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau} \cdot V_{\ell,b}^{\tau v_{y,\ell,b}}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$	successful decryption for $x > x^*$

For instance, the index-hiding security game of EIPL-IPFE (or EIPL-BE of [GKW19]) does not allow an adversary $\mathscr{A}$ to query a secret key for the challenge index i^*, however, it is not the same for EIPL-IBIPFE. In this case, $\mathscr{A}$ can ask for a secret key for a vector $\boldsymbol{u}$ associated to the tuple $(i^*, \mathsf{id}, \mathsf{gid} \neq \mathsf{gid}^*)$ where gid^* is the challenge group identity. Our idea is to introduce an additional randomness $\widehat{r}$ that is multiplied with the term $c_y = \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}$ when a secret key for $(i, \mathsf{id}, \mathsf{gid})$ is generated (as shown in the construction). Observe that the same randomness is used to randomize $\mathrm{H}(\mathsf{gid})$ while computing the secret key component K_2. This enables us to simulate the secret keys for tuples of the form $(i^*, \mathsf{id}, \mathsf{gid} \neq \mathsf{gid}^*)$ using the (modified) *Decisional 3-party Diffie-Hellman* assumption [BSW06]. The message-hiding security is proved utilizing the techniques used in our IBIPFE, however, we need to rely on the joint-D3DH assumption instead of the plain DBDH assumption to surpass the same difficulty faced in the case of EIPL-IPFE. We provide a detailed security analysis with all the five security notions of EIPL-IBIPFE in subsection 7.2.

2.7 EIPL-IBIPFE from LWE

We propose a generic construction of EIPL-IBIPFE from the LWE assumption. We use two primitives *attribute-based IPFE (ABIPFE)* [PD21, LLW21] and *mixed functional encryption (MFE)* [CVW+18] as the building blocks which have been built under the LWE assumption with different security levels.

The notion of ABIPFE is a more generalized version of IBIPFE. In ABIPFE, secret keys are generated with respect to a vector $\boldsymbol{u}$ and a policy $\mathscr{C}$, and ciphertexts are computed for a message vector $\boldsymbol{v}$ and an attribute att such that decryption recovers $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ if $\mathscr{C}(\mathsf{att}) = 0$. An MFE scheme can be viewed as a dual of standard FE in which a secret key is associated with a message M, and a ciphertext is associated with a (boolean) function f. Additionally, in MFE, there are two encryption algorithms – PK-Enc and SK-Enc. The *public* encryption PK-Enc encrypts the public parameters while the *secret* encryption SK-Enc encrypts the circuit f by the master secret key. The decryption algorithm in an MFE scheme outputs $f(M) \in \{0,1\}$.

We extend the framework of EIPL-BE [GKW19] to support inner product functionality by replacing the underlying ABE with an ABIPFE scheme [PD21, LLW21]. Additionally, we embed the group identity into the ABIPFE system to achieve our goal of EIPL-IBIPFE. We now describe our generic EIPL-IBIPFE construction. Let us consider two schemes such as $\mathsf{ABIPFE} = (\mathsf{Setup}_1, \mathsf{KeyGen}_1, \mathsf{Enc}_1, \mathsf{Dec}_1)$ and $\mathsf{MFE} = (\mathsf{Setup}_2, \mathsf{KeyGen}_2, \mathsf{PK\text{-}Enc}_2, \mathsf{SK\text{-}Enc}_2, \mathsf{Dec}_2)$ and let $\kappa = \log(nkk')$.

Setup($1^\lambda, n, 1^k, 1^{k'}, 1^m$): Computes $(\mathsf{msk}_1, \mathsf{mpk}_1) \leftarrow \mathsf{Setup}_1(1^\lambda, 1^m, 1^{k'})$; $(\mathsf{msk}_2, \mathsf{mpk}_2) \leftarrow \mathsf{Setup}_2(1^\lambda, 1^\kappa)$. Sets $\mathsf{mpk} = (\mathsf{mpk}_1, \mathsf{mpk}_2)$; $\mathsf{msk} = (\mathsf{msk}_1, \mathsf{msk}_2)$

KeyGen(msk, i, id, gid, $\boldsymbol{u}$): Computes $\mathsf{sk}_2 \leftarrow \mathsf{KeyGen}_2(\mathsf{msk}_2, (i, \mathsf{id}, \mathsf{gid}))$, $\mathsf{sk}_1 \leftarrow \mathsf{KeyGen}_1(\mathsf{msk}_1, \mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot), \boldsymbol{u})$ where

$$\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{ct}_2, \mathsf{gid}') = \begin{cases} 1 - \mathsf{Dec}_2(\mathsf{sk}_2, \mathsf{ct}_2), & \text{if } \mathsf{gid} = \mathsf{gid}' \\ 1, & \text{elsewhere} \end{cases}$$

Outputs $\mathsf{sk}_{\boldsymbol{u}} = \mathsf{sk}_1$

Enc(mpk, gid′, $\boldsymbol{v}$): $\mathsf{ct}_2 \leftarrow \mathsf{PK\text{-}Enc}_2(\mathsf{mpk}_2)$, $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_1(\mathsf{mpk}_1, (\mathsf{ct}_2, \mathsf{gid}'), \boldsymbol{v})$. Sets $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ct}_1$.

SplEnc(msk, gid′, $\boldsymbol{v}$, $(\boldsymbol{i^*}, \boldsymbol{\ell^*}, \boldsymbol{b^*})$): Compute $\mathsf{ct}_2 \leftarrow \mathsf{SK\text{-}Enc}_2(\mathsf{msk}_2, f_{i^*,\ell^*,b^*,\mathsf{gid}'})$ where

$$f_{i^*,\ell^*,b^*,\mathsf{gid}'}(j, \mathsf{id}, \mathsf{gid}) = \begin{cases} 1, & \text{if } (j \geq i^* + 1) \vee (i^*, \ell^*) = (j, \perp) \vee (i^*, \mathsf{id}_{\ell^*}) = \\ & \qquad (j, 1 - b^*) \wedge (\mathsf{gid} = \mathsf{gid}') \\ 0, & \text{otherwise} \end{cases}$$

and $\mathsf{ct}_1 \leftarrow \mathsf{Enc}_1(\mathsf{mpk}_1, (\mathsf{ct}_2, \mathsf{gid}'), \boldsymbol{v})$. Outputs $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ct}_1$.

Dec($\mathsf{ct}_{\boldsymbol{v}}$, $\mathsf{sk}_{\boldsymbol{u}}$): $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \mathsf{Dec}_1(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}})$.

Relying on the security of ABIPFE and MFE, we prove that our EIPL-IBIPFE is selectively secure. The complete security analysis is provided in subsection 8.2.

Instantiation using ABIPFE of [LLW21] and MFE of [GKW18]. While the work of Goyal et al. [GKW18] proposes a construction of an MFE scheme for log-depth circuits which is sufficient for our EIPL-IBIPFE, but the (most suitable) ABIPFE of Lai et al. [LLW21] is proven secure with only a bounded number of *accepting* keys. Furthermore, the ciphertext size grows additively with the number of accepting keys[3]. By accepting keys we mean the secret keys that can successfully decrypt the challenge ciphertext. Interpreting it into our setting, this indeed implies that our EIPL-IBIPFE has a bounded number of users (i.e., the parameter n) and the ciphertext size grows additively with n and k. This eventually prevents us to allow an exponential number of users in the system, and thus the complexity leveraging technique of [GKW19] can not be employed to eliminate the indices i from EI-TIBIPFE. Although complexity leveraging techniques are common in literature, it incurs a considerable amount of security loss in the system and hence may not be a practical solution for dropping the indices. Lastly, we *emphasize* that it is not at all a drawback of our transformation, rather a limitation of existing ABIPFEs [PD21, LLW21]. If one can achieve an ABIPFE scheme with $\log n$ size ciphertexts in future, then our EIPL-IBIPFE will yield an EI-TIBIPFE having optimal size parameters.

3 Preliminaries

Notations: Let λ be the security parameter that belongs to the set of natural numbers, 1^λ denotes its unary representation, and $\mathsf{poly}(\lambda)$ be a polynomial in λ. For a prime p, let $\mathbb{Z}_p$ denotes the field $\mathbb{Z}/p\mathbb{Z}$. For a set S, we use the notation $s \leftarrow S$ to indicate the fact that s is

[3] The ABIPFE of [PD21] supports a single accepting key, so it can not be used in our transformation.

sampled uniformly at random from a finite set S. We write $x \leftarrow \mathscr{X}$ to denote that the element x is sampled at random according to the distribution $\mathscr{X}$. For any natural number n, $[n]$ denotes the set $\{1,2,\ldots,n\}$. We use a bold lower-case letter e.g., $\boldsymbol{a}$ to denote a vector, and a bold upper-case letter e.g., $\mathbf{A}$ denotes a matrix. The i-th element of the vector $\boldsymbol{a}$ is expressed as a_i, and (i,j)-th element of a matrix $\mathbf{A}$ is represented by $a_{i,j}$. The transpose of a matrix $\mathbf{A}$ is denoted by $\mathbf{A}^\top$. Let $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^m$, then the inner product between the vectors is defined as $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{m} u_i v_i \in \mathbb{Z}$. A function $\mathsf{negl} : \mathbb{N} \to \mathbb{R}$ is said to be a negligible function if $\mathsf{negl}(\lambda) = \lambda^{-\omega(1)}$. An algorithm A is said to be a probabilistic polynomial time (PPT) algorithm if it is modeled as a probabilistic Turing machine that runs in time $\mathsf{poly}(\lambda)$. If for any PPT adversary $\mathscr{A}$ such that $|\Pr[\mathscr{A}(1^\lambda, X) = 1] - \Pr[\mathscr{A}(1^\lambda, Y) = 1]|$ is *negligible* in λ, then we say that the two distributions are indistinguishable, denoted by $X \approx Y$.

3.1 Bilinear Group

A bilinear group $\mathbb{BG} = (p, q, N = p \cdot q, \mathbb{G}, \mathbb{G}_T, g, e(\cdot,\cdot))$ consists of the two primes p, q, two multiplicative (source and target) groups $\mathbb{G}, \mathbb{G}_T$ (respectively) with the order $|\mathbb{G}| = |\mathbb{G}_T| = N$, g as the generator of the group $\mathbb{G}$ and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. It satisfies the following:

- *bilinearity:* $e(g^a, g^b) = e(g,g)^{ab}$ for all $g \in \mathbb{G},\ a, b \in \mathbb{Z}_N$ and
- *non-degeneracy:* $e(g,g)$ is a generator of $\mathbb{G}_T$.

A bilinear group generator $\mathscr{G}_{\mathsf{BG.Gen}}(1^\lambda)$ takes the security parameter λ and outputs a bilinear group $\mathbb{BG} = (p, q, N, \mathbb{G}, \mathbb{G}_T, g, e(\cdot,\cdot))$ with a λ-bit composite integer $N = p \cdot q$. We consider $\mathbb{G}_p$ and $\mathbb{G}_q$ as the subgroups of $\mathbb{G}$ and their orders p and q respectively.

3.2 Complexity Assumptions

Let $\mathbb{BG} = (p, q, N, \mathbb{G}, \mathbb{G}_T, g, e(\cdot,\cdot)) \leftarrow \mathscr{G}_{\mathsf{BG.Gen}}(1^\lambda)$ be a bilinear group with composite order $N = p \cdot q$. We define a series of source and target group-based assumptions [BSW06, GKW19].

Assumption 1 (Modified-1 Decisional 3-party Diffie-Hellman Assumption) *(*modified-1 *D3DH) For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\left(\begin{array}{c}\mathbb{BG}, g_p, g_q, g_p^a, g_p^b,\\ g_p^c, g_p^{b^2}, g_p^{b^3}, g_p^{b^4}\\ g_p^{b^2 c}, g_p^{b^3 c}, T_{\mathfrak{b}}\end{array}\right) = \mathfrak{b} : \begin{array}{c}\mathbb{BG}; g_p \leftarrow \mathbb{G}_p; g_q \leftarrow \mathbb{G}_q; a, b, c, r \leftarrow \mathbb{Z}_p;\\ T_0 = g_p^{abc}; T_1 = g_p^r; \mathfrak{b} \leftarrow \{0,1\}\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Assumption 2 ((Plain) Decisional Bilinear Diffie-Hellman Assumption) *(DBDH) For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\left(\begin{array}{c}\mathbb{BG}, g, g^a, g^b,\\ g^c, T_{\mathfrak{b}}\end{array}\right) = \mathfrak{b} : \begin{array}{c}\mathbb{BG}; g \leftarrow \mathbb{G}; a, b, c, r \leftarrow \mathbb{Z}_N;\\ T_0 = e(g,g)^{abc}; T_1 = e(g,g)^r; \mathfrak{b} \leftarrow \{0,1\}\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Assumption 3 (Modified-2 Decisional 3-party Diffie-Hellman Assumption) *(*modified-2 *D3DH) For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\left(\begin{array}{c}\mathbb{BG}, g_p, g_q, g_p^a, g_p^b,\\ g_p^c, g_p^{b^2}, T_{\mathfrak{b}}\end{array}\right) = \mathfrak{b} : \begin{array}{c}\mathbb{BG}; g_p \leftarrow \mathbb{G}_p; g_q \leftarrow \mathbb{G}_q; a, b, c, r \leftarrow \mathbb{Z}_p;\\ T_0 = g_p^{abc}; T_1 = g_p^r; \mathfrak{b} \leftarrow \{0,1\}\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Assumption 4 (Diffie-Hellman Sub-group Decisional Assumption) (DHSD) *For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\begin{pmatrix}\mathbb{BG}, g, h, g_p, g_q, g_q^a, h_q^a,\\ g^b g_p^c, h^b, T_{\mathfrak{b}}\end{pmatrix} = \mathfrak{b} : \begin{matrix}\mathbb{BG}; g = g_p g_q; h = h_p h_q;\\ g_p, h_p \leftarrow \mathbb{G}_p; g_q, h_q \leftarrow \mathbb{G}_q;\\ a, b, c \leftarrow \mathbb{Z}_N;\\ T_0 \leftarrow \mathbb{G}_q; T_1 \leftarrow \mathbb{G}; \mathfrak{b} \leftarrow \{0,1\}\end{matrix}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Assumption 5 (Bilinear Sub-group Decisional Assumption) (BSD) *For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\begin{pmatrix}\mathbb{BG}, g, g_p, g_q,\\ e(T_{\mathfrak{b}}, g)\end{pmatrix} = \mathfrak{b} : \begin{matrix}\mathbb{BG}; g = g_p g_q; h = h_p h_q;\\ g_p \leftarrow \mathbb{G}_p; g_q \leftarrow \mathbb{G}_q; g \leftarrow \mathbb{G}\\ T_0 \leftarrow \mathbb{G}_p; T_1 \leftarrow \mathbb{G}; \mathfrak{b} \leftarrow \{0,1\}\end{matrix}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Assumption 6 (Relaxed 3-party Diffie-Hellman Assumption) (R3DH) *For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\begin{pmatrix}\mathbb{BG}, g, g_p, g_q,\\ g_q^a, g_p^{\widetilde{a}} g_q^{a^2}, g_p^{\widetilde{a}\widetilde{c}}, g_p^{\widetilde{c}} g_q^c, T_{\mathfrak{b}}\end{pmatrix} = \mathfrak{b} : \begin{matrix}\mathbb{BG}; g_p \leftarrow \mathbb{G}_p; g_q \leftarrow \mathbb{G}_q;\\ \widetilde{a}, \widetilde{c} \leftarrow \mathbb{Z}_p; a, c \leftarrow \mathbb{Z}_q;\\ T_0 = g_q^{a^2 c}; T_1 \leftarrow \mathbb{G}; \mathfrak{b} \leftarrow \{0,1\}\end{matrix}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Assumption 7 (Joint Decisional 3-party Diffie-Hellman Assumption) (joint D3DH) *For every PPT adversary $\mathscr{A}$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathscr{A}\begin{pmatrix}\mathbb{BG}, g_p, g_q, g_p^a, g_p^b,\\ g_p^c, g_q^a, g_q^b, g_q^c, (T_{\mathfrak{b}}, S_{\mathfrak{b}})\end{pmatrix} = \mathfrak{b} : \begin{matrix}\mathbb{BG}; g_p \leftarrow \mathbb{G}_p; g_q \leftarrow \mathbb{G}_q; a, b, c, r \leftarrow \mathbb{Z}_N;\\ T_0 = g_p^{abc}; T_1 = g_p^r;\\ S_0 = g_q^{abc}; S_1 = g_q^r; \mathfrak{b} \leftarrow \{0,1\}\end{matrix}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

3.3 Identity-Based Inner Product Functional Encryption

An identity-based inner product functional encryption (IBIPFE) scheme consists of four PPT algorithms IBIPFE = (Setup, KeyGen, Enc, Dec) which works as follows:

Setup$(1^\lambda, 1^k, 1^m) \to$ (msk, mpk): On input the security parameter λ, a length k of identities and a vector length m (as unary), the trusted authority generates a master secret key msk and a master public key mpk.

KeyGen(msk, id, $\boldsymbol{u}$) $\to \mathsf{sk}_{\boldsymbol{u}}$: The trusted authority takes as input the master secret key msk, an identity $\mathsf{id} \in \{0,1\}^k$, a vector $\boldsymbol{u} \in \mathbb{Z}^m$, and outputs a secret key $\mathsf{sk}_{\boldsymbol{u}}$.

Enc(mpk, id$'$, $\boldsymbol{v}$) $\to \mathsf{ct}_{\boldsymbol{v}}$: The encryption algorithm takes as input the master public key mpk, an identity $\mathsf{id}' \in \{0,1\}^k$ and a message vector $\boldsymbol{v} \in \mathbb{Z}^m$. It outputs a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$.

$\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) \to \zeta/\perp$: The decryption algorithm uses a secret key $\mathsf{sk}_{\boldsymbol{u}}$ to decrypt the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$. It either outputs a decrypted value ζ on successful decryption or a symbol $\perp$ indicating decryption failure.

Correctness. An IBIPFE = (Setup, KeyGen, Enc, Dec) scheme is said to be correct if for all $\lambda, k, m \in \mathbb{N}$, $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^m$, identity $\mathsf{id} \in \{0,1\}^k$, there exists a *negligible* function negl such that the following holds

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v}\rangle : \begin{array}{r}(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^m) \\ \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}, \boldsymbol{u}) \\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}, \boldsymbol{v})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is taken over the random coins of Setup, KeyGen and Enc of IBIPFE.

Definition 1 (Adaptive security of IBIPFE) *An IBIPFE scheme is said to satisfy an adaptive indistinguishability-based (Adp-IND-CPA) security if for every PPT adversary $\mathscr{A}$, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{r}(\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^m) \\ (\mathsf{id}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \boldsymbol{v}^{(\mathfrak{b})})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following restriction on the key generation oracle.

- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of the form* $(\mathsf{id}, \boldsymbol{u})$ *satisfying the condition* $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)}\rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)}\rangle$ *if* $\mathsf{id} = \mathsf{id}^*$.

Definition 2 (Selective security of IBIPFE) *An IBIPFE scheme is said to satisfy the selective indistinguishability-based (Sel-IND-CPA) security if for every PPT adversary $\mathscr{A}$, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$, the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{r}(\mathsf{id}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k, 1^m) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{id}^*, \boldsymbol{v}^{(\mathfrak{b})})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following restriction on the key generation oracle.

- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of the form* $(\mathsf{id}, \boldsymbol{u})$ *satisfying the condition* $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)}\rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)}\rangle$ *if* $\mathsf{id} = \mathsf{id}^*$.

Remark 1 (IPFE) *If we omit the identity from the above syntax of IBIPFE, then it yields the primitive of inner product functional encryption (IPFE).*

3.4 Attribute-Based Inner Product Functional Encryption

An attribute-based inner product functional encryption (ABIPFE) scheme for a class of functions $\mathscr{F}_\lambda = \{\mathscr{C} : \mathscr{S}_\lambda \to \{0,1\}\}$, a predicate space $\mathscr{X}_\lambda$ and a message space $\mathscr{Y}_\lambda$ consists of four PPT algorithms ABIPFE = (Setup, KeyGen, Enc, Dec) and details about these algorithms are given below.

Setup($1^\lambda, 1^m$) → (msk, mpk): On input the security parameter λ, a vector length m (as unary), the trusted authority generates a master secret key msk and a master public key mpk.

KeyGen(msk, $\mathscr{C}$, $\boldsymbol{u}$) → $\mathsf{sk}_{\boldsymbol{u}}$: The trusted authority takes input the master secret msk, a function $\mathscr{C} \in \mathscr{F}_\lambda$, a vector $\boldsymbol{u} \in \mathbb{Z}^m$ and outputs a secret key $\mathsf{sk}_{\boldsymbol{u}}$.

Enc(mpk, att, $\boldsymbol{v}$) → $\mathsf{ct}_{\boldsymbol{v}}$: This encryption algorithm takes as input the master public key mpk, an attribute $\mathsf{att} \in \mathscr{S}_\lambda$ and a message vector $\boldsymbol{v} \in \mathbb{Z}^m$. It outputs a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$.

Dec($\mathsf{sk}_{\boldsymbol{u}}$, $\mathsf{ct}_{\boldsymbol{v}}$) → $\zeta/\perp$: The decryption algorithm uses a secret key $\mathsf{sk}_{\boldsymbol{u}}$ to decrypts the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$. It either outputs a decrypted value ζ on successful decryption or a symbol $\perp$ indicating failure.

Correctness. An ABIPFE = (Setup, KeyGen, Enc, Dec) scheme is said to be correct if for all $\lambda, m \in \mathbb{N}$, $\mathscr{C} \in \mathscr{F}_\lambda, \mathsf{att} \in \mathscr{S}_\lambda, \boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^m$, there exists a *negligible* function negl such that the following holds,

$$\Pr\left[\begin{array}{r} (\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^m) \\ \mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle \wedge \mathscr{C}(\mathsf{att}) = 0 : \quad \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathscr{C}, \boldsymbol{u}) \\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{att}, \boldsymbol{v}) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

where the probability is taken over the random coins of Setup, KeyGen and Enc of ABIPFE as described above.

Definition 3 (IND-CPA Security) *An ABIPFE scheme is said to satisfy IND-CPA security if for every PPT adversary $\mathscr{A}$, there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\begin{array}{r} (\mathsf{msk}, \mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^m) \\ \mathscr{A}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : (\mathsf{att}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk}, \cdot, \cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \boldsymbol{v}^{(\mathfrak{b})}) \end{array}\right] \leq \frac{1}{2} + \mathit{negl}(\lambda)$$

with the following restriction on the key generation oracle.

- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of the form $(\mathscr{C}, \boldsymbol{u})$ satisfying the condition $\langle \boldsymbol{u}_i, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}_i, \boldsymbol{v}^{(1)} \rangle$ if $\mathscr{C}(\mathsf{att}^*) = 0$.*

3.5 Mixed Functional Encryption

A mixed functional encryption (MFE) scheme for a class of functions $\mathscr{F} = \{\mathscr{F}_k\}_k$ and message spaces $\mathscr{M} = \{\mathscr{M}_k\}_k$ where $f : \mathscr{M}_k \to \{0,1\}$ consists of five PPT algorithms MFE = (Setup, KeyGen, PK-Enc, SK-Enc, Dec) and details about these algorithms are given below.

Setup($1^\lambda, 1^k$) → (msk, mpk): On input the security parameter λ as unary and a functionality index k, the trusted authority outputs a master secret key msk and a master public key mpk.

KeyGen(msk, M) → sk_M: On input the master secret key msk, a message $M \in \mathscr{M}$, the trusted authority generates the secret key sk_M.

PK-Enc(mpk) → ct: This normal public key encryption algorithm takes input the master public key mpk and outputs a ciphertext ct.

SK-Enc(msk, f) → ct: The secret encryption algorithm takes input the master secret key msk, a function $f \in \mathscr{F}_k$ and it generates the ciphertext ct.

Dec(sk_M, ct) → {0, 1}: On input the secret key sk_M, a ciphertext ct, the decryptor outputs a single bit between 0 and 1.

Correctness. A MFE = (Setup, KeyGen, PK-Enc, SK-Enc, Dec) is said to be *correct* if for all $\lambda, k \in \mathbb{N}$, $M \in \mathscr{M}$, there exists negligible functions negl_1, negl_2 such that the following holds,

$$\Pr\left[\begin{array}{c} (\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ \mathsf{Dec}(\mathsf{sk}_M, \mathsf{ct}) = 1 : \quad \mathsf{sk}_M \leftarrow \mathsf{KeyGen}(\mathsf{msk}, M) \\ \mathsf{ct} \leftarrow \mathsf{PK\text{-}Enc}(\mathsf{mpk}) \end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda),$$

$$\Pr\left[\begin{array}{c} (\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ \mathsf{Dec}(\mathsf{sk}_M, \mathsf{ct}) = f(M) : \quad \mathsf{sk}_M \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \boldsymbol{u}) \\ \mathsf{ct} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{msk}, f) \end{array}\right] \geq 1 - \mathsf{negl}_2(\lambda)$$

where the probability is taken over the random coins of Setup, KeyGen, PK-Enc and SK-Enc of MFE as described above.

Definition 4 (q-bounded Restricted Function IND (F-IND)) *Let $q(\cdot)$ be a fixed polynomial. A MFE scheme is said to be q-bounded restricted function indistinguishability (F-IND) security if for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\begin{array}{c} (1^k, f^{(0)}, f^{(1)}) \leftarrow \mathscr{A}(1^\lambda) \\ \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)}(\mathsf{ct}_\mathfrak{b}) = \mathfrak{b} : \quad (\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_\mathfrak{b} \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{mpk}, f^{(\mathfrak{b})}) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then a MFE is said to satisfy F-IND security:

- *$\mathscr{A}$ can make at most q queries to* $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)$ *oracle.*
- *All queries M of $\mathscr{A}$ to* $\mathsf{KeyGen}(\mathsf{msk},\cdot)$ *oracle should satisfy $f^{(0)}(M) = f^{(1)}(M)$.*
- *$\mathscr{A}$ must make all (at most q)* $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)$ *oracle queries before making any query to* $\mathsf{KeyGen}(\mathsf{msk},\cdot)$ *oracle.*

Definition 5 (q-bounded Restricted Accept IND (A-IND)) *Let $q(\cdot)$ be a fixed polynomial. A MFE scheme is said to be q-bounded restricted accept indistinguishability (A-IND) security if for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every*

$\lambda \in \mathbb{N}$ *the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot),\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)}(\mathsf{ct}_{\mathfrak{b}}) = \mathfrak{b} : \begin{array}{c}(1^k, f^{(*)}) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, 1^k) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_0 \leftarrow \mathsf{PK\text{-}Enc}(\mathsf{mpk}) \\ \mathsf{ct}_1 \leftarrow \mathsf{SK\text{-}Enc}(\mathsf{mpk}, f^{(*)})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then a MFE is said to satisfy F-IND security:

- $\mathscr{A}$ *can make at most q queries to* $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)$ *oracle.*
- *All queries M of* $\mathscr{A}$ *to* $\mathsf{KeyGen}(\mathsf{msk},\cdot)$ *oracle should satisfy* $f^{(*)}(M) = 1$ *and* $f(M) = 1$ *for every query f to* $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)$.
- $\mathscr{A}$ *must make all (at most q)* $\mathsf{SK\text{-}Enc}(\mathsf{msk},\cdot)$ *oracle queries before making any query to* $\mathsf{KeyGen}(\mathsf{msk},\cdot)$ *Oracle.*

3.6 Embedded Identity Private Linear Inner Product Functional Encryption

An EIPL-IPFE for a message vector space $\mathscr{Y} = \{\mathscr{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, an identity space $\mathscr{ID} = \{\{0,1\}^k : k \in \mathbb{N}\}$ consists of five PPT algorithms EIPL-IPFE = (Setup, KeyGen, Enc, SplEnc, Dec) and details about these algorithms are given below.

Setup$(1^\lambda, n, 1^k, 1^m) \rightarrow$ (msk, mpk, key): The trusted authority takes as input the security parameter λ, the index space n, the 'identity space' parameter k, a vector length parameter m, and outputs a master secret key msk, a master public key mpk and a key key. The master public key mpk and the key key are made public while the master secret key msk is kept secret to the trusted authority.

KeyGen(msk, i, id, $\boldsymbol{u}$) $\rightarrow \mathsf{sk}_{\boldsymbol{u}}$: On input the master secret key msk, an index $i \in [n]$, an identity $\mathsf{id} \in \{0,1\}^k$ and a vector $\boldsymbol{u} \in \mathbb{Z}^m$, the trusted authority outputs a secret key $\mathsf{sk}_{\boldsymbol{u}}$.

Enc(mpk, $\boldsymbol{v}$) $\rightarrow \mathsf{ct}_{\boldsymbol{v}}$: This algorithm is run by an encryptor by taking input as mpk, a message vector $\boldsymbol{v} \in \mathbb{Z}^m$ and generates a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ associated to the vector $\boldsymbol{v}$.

SplEnc(key, $\boldsymbol{v}$, (i, ℓ, b)) $\rightarrow \mathsf{ct}_{\boldsymbol{v}}$: This algorithm outputs a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ by taking input as key, a message vector $\boldsymbol{v} \in \mathbb{Z}^m$ and index-position-bit tuple $(i, \ell, b) \in [n+1] \times ([k] \cup \{\perp\}) \times \{0,1\}$. If key = mpk then EIPL-IPFE is called public key EIPL-IPFE, else called private key EIPL-IPFE.

Dec($\mathsf{sk}_{\boldsymbol{u}}$, $\mathsf{ct}_{\boldsymbol{v}}$) $\rightarrow \zeta/\perp$: On input the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ and the secret key $\mathsf{sk}_{\boldsymbol{u}}$, the decryptor outputs a decrypted value ζ or a symbol $\perp$ indicating failure.

Correctness. An EIPL-IPFE = (Setup, KeyGen, Enc, SplEnc, Dec) scheme is said to be correct if there exists *negligible* functions negl_1, negl_2 such that for all $\lambda, n, k, m \in \mathbb{N}$, $\boldsymbol{v} \in \mathbb{Z}^m$, $i \in [n+1]$, $j \in [n]$, an identity $\mathsf{id} \in \{0,1\}^k$, $\ell \in ([k] \cup \{\perp\})$, $b \in \{0,1\}$, the following holds.

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v}\rangle : \begin{array}{c}(\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^m) \\ \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, j, \mathsf{id}, \boldsymbol{u}) \\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \boldsymbol{v})\end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda),$$

If $(j \geq i+1) \vee (i,\ell) = (j,\perp) \vee (i,\mathsf{id}_\ell) = (j,1-b)$, then the following property holds,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}},\mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u},\boldsymbol{v}\rangle : \begin{array}{r}(\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda,n,1^k,1^m)\\ \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk},j,\mathsf{id},\boldsymbol{u})\\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{SplEnc}(\mathsf{mpk},\boldsymbol{v},(i,\ell,b))\end{array}\right] \geq 1-\mathsf{negl}_2(\lambda),$$

3.7 IND-CPA security of EIPL-IPFE

Let $q(\cdot)$ be a fixed polynomial. The security definitions for EIPL-IPFE are a generalization from the q-query security notions of EIPL-BE [GKW19], which we discuss below.

Definition 6 (q-query Normal Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r}(1^n,1^k,1^m) \leftarrow \mathscr{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda,n,1^k,1^m)\\ \boldsymbol{v} \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot,\cdot)}(\mathsf{mpk})\\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{Enc}(\mathsf{mpk},\boldsymbol{v})\\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\boldsymbol{v},(1,\perp,0))\end{array}\right] \leq \frac{1}{2}+\mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IPFE scheme is said to satisfy q-query normal hiding security:

- SplEnc *Oracle: $\mathscr{A}$ can make at most $q(\lambda)$ queries of the form $(\boldsymbol{v},(1,\ell,\gamma))$.*
- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of distinct indices. That is, if $\mathscr{A}$ makes the queries $(i_1,\mathsf{id}_1,\boldsymbol{u}_1),(i_2,\mathsf{id}_2,\boldsymbol{u}_2),\ldots,(i_\kappa,\mathsf{id}_\kappa,\boldsymbol{u}_\kappa)$, then $i_a \neq i_b$ when $a \neq b$ for every $a,b \in [\kappa]$.*

Definition 7 (q-query Index Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r}(1^n,1^k,1^m,i^*) \leftarrow \mathscr{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda,n,1^k,1^m)\\ \boldsymbol{v} \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk})\\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\boldsymbol{v},(i^*,\perp,0))\\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\boldsymbol{v},(i^*+1,\perp,0))\end{array}\right] \leq \frac{1}{2}+\mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IPFE scheme is said to satisfy q-query index hiding security:

- SplEnc *Oracle: $\mathscr{A}$ can make at most $q(\lambda)$ queries of the form $(\boldsymbol{v},(i,\ell,\gamma))$, where the index i must be equal to either i^* or i^*+1.*
- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of distinct indices and should not be of the form $(i^*,\mathsf{id},\boldsymbol{u})$. That is, if $\mathscr{A}$ makes the key queries $(i_1,\mathsf{id}_1,\boldsymbol{u}_1),(i_2,\mathsf{id}_2,\boldsymbol{u}_2),\ldots,(i_\kappa,\mathsf{id}_\kappa,\boldsymbol{u}_\kappa)$, then $i_a \neq i_b$ when $a \neq b$ for every $a,b \in [\kappa]$ and $i_a \neq i^*$ for every $a \in [\kappa]$.*

Definition 8 (q-query Lower Identity Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r} (1^n, 1^k, 1^{k'}, 1^m, i^*, \ell^*, b^*) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^m) \\ \boldsymbol{v} \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \boldsymbol{v}, (i^*, \perp, 0)) \\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \boldsymbol{v}, (i^*, \ell^*, b^*)) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IPFE scheme is said to satisfy q-query lower identity hiding security:

- SplEnc *Oracle: $\mathscr{A}$ can make at most $q(\lambda)$ queries of the form $(\boldsymbol{v}, (i, \ell, \gamma))$, where the index i must be equal to i^*.*
- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of distinct indices and should not be of the form $(i^*, \mathsf{id}, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = b^*$. That is, if $\mathscr{A}$ makes the key queries $(i_1, \mathsf{id}_1, \boldsymbol{u}_1)$, $(i_2, \mathsf{id}_2, \boldsymbol{u}_2)$, $\ldots$, $(i_\kappa, \mathsf{id}_\kappa, \boldsymbol{u}_\kappa)$, then $i_a \neq i_b$ when $a \neq b$ for every $a, b \in [\kappa]$ and $i_a \neq i^*$ or $(\mathsf{id}_a)_{\ell^*} \neq b^*$ for every $a \in [\kappa]$.*

Definition 9 (q-query Upper Identity Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r} (1^n, 1^k, 1^m, i^*, \ell^*, b^*) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^m) \\ \boldsymbol{v} \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \boldsymbol{v}, (i^*+1, \perp, 0)) \\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \boldsymbol{v}, (i^*, \ell^*, b^*)) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IPFE scheme is said to satisfy q-query upper identity hiding security:

- SplEnc *Oracle: $\mathscr{A}$ can make at most $q(\lambda)$ queries of the form $(\boldsymbol{v}, (i, \ell, \gamma))$, where the index i must be equal to either i^* or $i^* + 1$.*
- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of distinct indices and should not be of the form $(i^*, \mathsf{id}, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = 1 - b^*$. That is, if $\mathscr{A}$ makes the key queries $(i_1, \mathsf{id}_1, \boldsymbol{u}_1)$, $(i_2, \mathsf{id}_2, \boldsymbol{u}_2)$, $\ldots$, $(i_\kappa, \mathsf{id}_\kappa, \boldsymbol{u}_\kappa)$, then $i_a \neq i_b$ when $a \neq b$ for every $a, b \in [\kappa]$ and $i_a \neq i^*$ or $(\mathsf{id}_a)_{\ell^*} \neq 1 - b^*$ for every $a \in [\kappa]$.*

Definition 10 (q-query Message Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{r} (1^n, 1^k, 1^m, i^*) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^m) \\ (\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{SplEnc}(\mathsf{key}, , \boldsymbol{v}^{(\mathfrak{b})}, (i^*, \perp, 0)) \end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IPFE scheme is said to satisfy q-query message hiding security:

– SplEnc *Oracle:* $\mathscr{A}$ *can make at most* $q(\lambda)$ *queries of the form* $(\boldsymbol{v},(i,\ell,\gamma))$*, where the index* i *must be equal to* i^*.
– KeyGen *Oracle: All queries of* $\mathscr{A}$ *should be of distinct indices and the form of* $(i,\mathsf{id},\boldsymbol{u})$ *for* $i \geq i^*$ *satisfying the condition* $\langle \boldsymbol{u},\boldsymbol{v}^{(0)}\rangle = \langle \boldsymbol{u},\boldsymbol{v}^{(1)}\rangle$*. That is, if* $\mathscr{A}$ *makes the key queries* $(i_1,\mathsf{id}_1,\boldsymbol{u}_1)$, $(i_2,\mathsf{id}_2,\boldsymbol{u}_2)$, …, $(i_\kappa,\mathsf{id}_\kappa,\boldsymbol{u}_\kappa)$*, then* $i_a \neq i_b$ *when* $a \neq b$ *for every* $a,b \in [\kappa]$*, and if* $i_a \geq i^*$ *then* $\langle \boldsymbol{u},\boldsymbol{v}^{(0)}\rangle = \langle \boldsymbol{u},\boldsymbol{v}^{(1)}\rangle$ *for any* $a \in [\kappa]$.

3.8 Embedded Identity Traceable Identity-Based Inner Product Functional Encryption

An embedded identity traceable identity-based inner product functional encryption (EI-TIBIPFE) scheme consists of five PPT algorithms EI-TIBIPFE = (Setup, KeyGen, Enc, Dec, Trace), working as follows:

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m) \to (\mathsf{msk},\mathsf{mpk},\mathsf{key})$: The trusted authority takes as input the security parameter λ, an index n, a *user identity space* parameter k, a *group identity space* parameter k', a vector length parameter m and generates the master secret key msk, a master public key mpk and, a tracing key key.

KeyGen$(\mathsf{msk}, i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u}) \to \mathsf{sk}_{\boldsymbol{u}}$: On input the master secret key msk, user index $i \in [n]$, a user identity $\mathsf{id} \in \{0,1\}^k$, a group identity $\mathsf{gid} \in \{0,1\}^{k'}$ and a vector $\boldsymbol{u} \in \mathbb{Z}^m$, the trusted authority outputs a secret key $\mathsf{sk}_{\boldsymbol{u}}$.

Enc$(\mathsf{mpk}, \mathsf{gid}', \boldsymbol{v}) \to \mathsf{ct}_{\boldsymbol{v}}$: The encryption algorithm takes input the master public key mpk, a group identity $\mathsf{gid}' \in \{0,1\}^{k'}$, a message vector $\boldsymbol{v} \in \mathbb{Z}^m$, and produces a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$.

Dec$(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) \to \zeta/\perp$: The decryption algorithm is run by taking input a secret key $\mathsf{sk}_{\boldsymbol{u}}$ and a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$. It either outputs a message ζ on successful decryption or a symbol $\perp$ indicating decryption failure.

Trace$^{\mathscr{D}_{\boldsymbol{u}}}(\mathsf{key}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{gid}, \boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \to \boldsymbol{T}$: This algorithm has oracle access to a program $\mathscr{D}_{\boldsymbol{u}}$ associated with the vector $\boldsymbol{u}$, it takes as input the tracing key key, a group identity gid, the predicate vector $\boldsymbol{u}$, two message vectors $\boldsymbol{v}^{(0)}$, $\boldsymbol{v}^{(1)}$ and outputs a set of identities $T \subseteq \{0,1\}^k$. We call the tracing as public or private depending on whether key is equal to mpk or it is kept secret.

Correctness. An EI-TIBIPFE = (Setup, KeyGen, Enc, Dec, Trace) scheme is said to be correct if for all $\lambda, n, k, k', n \in \mathbb{N}$, $i \in [n]$, $\mathsf{id} \in \{0,1\}^k$, $\mathsf{gid} \in \{0,1\}^{k'}$ and $\boldsymbol{v}, \boldsymbol{u} \in \mathbb{Z}^m$, there exists a *negligible* function negl satisfying $\mathsf{gid} = \mathsf{gid}'$ such that the following holds,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}},\mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u},\boldsymbol{v}\rangle : \begin{array}{c}(\mathsf{msk},\mathsf{mpk}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m) \\ \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u}) \\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{gid}, \boldsymbol{v})\end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

where the probability is taken over the random coins of Setup, KeyGen and Enc of EI-TIBIPFE.

Definition 11 (Adaptive Security of EI-TIBIPFE) *An EI-TIBIPFE is said to satisfy adaptive indistinguishable-based* (Adp-IND-CPA) *security if for every PPT adversary* $\mathscr{A}$*, there exists a*

negligible function negl such that for every $\lambda \in \mathbb{N}$ the following holds:

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{r}(\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m) \\ (\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \mathsf{gid}^*) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}^{(\mathfrak{b})})\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

with the following restriction on the key generation oracle.

- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of form $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ with $i \in [n]$, $\mathsf{id} \in \{0,1\}^k, \mathsf{gid} \in \{0,1\}^{k'}$ and if $\mathsf{gid} = \mathsf{gid}^*$ then $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$ holds.*

We can similarly define the Sel-IND-CPA security of EI-TIBIPFE (alike to Definition 2) where the adversary submits the challenge tuple $(\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \mathsf{gid}^*)$ before it receives the public parameters.

Definition 12 (Security of Tracing) *For any non-negligible function $\epsilon(\cdot)$, polynomial $p(\cdot)$ and for all PPT adversary $\mathscr{A}$, consider the experiment $\mathsf{Expt}_{\mathscr{A}}^{\mathsf{EI\text{-}TIBIPFE}}(1^\lambda, \mathfrak{b})$ defined in Fig. 1. The tracing security of the scheme EI-TIBIPFE = (Setup, KeyGen, Enc, Dec, Trace) is defined as follows:*

1. $(1^n, 1^k, 1^{k'}, 1^m) \leftarrow \mathscr{A}(1^\lambda)$
2. $(\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m)$
3. $(\mathscr{D}_{\boldsymbol{u}}, \boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, \mathsf{gid}^*) \leftarrow \mathscr{A}^{O(\cdot)}$
4. $T \leftarrow \mathsf{Trace}^{\mathscr{D}_{\boldsymbol{u}}}(\mathsf{key}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{gid}^*, \boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$

The oracle $O(\cdot)$ has the msk hardwired in it and on query $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ the oracle runs $\mathsf{KeyGen}(\mathsf{msk}, i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ and sends the output iff the index i was not queried before, otherwise it sends $\perp$. Let $\mathscr{S}_{\mathsf{ID}}^{\boldsymbol{u}}$ be the set of all users identities (id's) queried by $\mathscr{A}$ associated with the vector $\boldsymbol{u}$. The above model defines the *adaptive* tracing security. In case of *selective* tracing, $\mathscr{A}$ selects $(\mathsf{gid}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$ before setup and it outputs the decoder $\mathscr{D}_{\boldsymbol{u}}$ after it queries some secret keys.

Fig. 1: $\mathsf{Expt}_{\mathscr{A}}^{\mathsf{EI\text{-}TIBIPFE}}(1^\lambda, \mathfrak{b})$

Based on the above experiment in Fig. 1, we define the following events and corresponding probabilities.

- Good-Decoder*:* $\Pr[\mathscr{D}_{\boldsymbol{u}}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \mathfrak{b} \leftarrow \{0,1\}, \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}^{(\mathfrak{b})})] \geq \frac{1}{2} + \epsilon(\lambda)$
  $\Pr\text{-G-D}_{\mathscr{A},\epsilon,p}(\lambda) = \Pr[\text{Good-Decoder} \wedge p(\lambda) \geq |\mathscr{S}_{\mathsf{ID}}^{\boldsymbol{u}}|]$.
- Cor-Tr*:* $T \neq \phi \wedge T \subset \mathscr{S}_{\mathsf{ID}}^{\boldsymbol{u}}$
  $\Pr\text{-Cor-Tr}_{\mathscr{A},\epsilon,p}(\lambda) = \Pr[\text{Cor-Tr}]$.
- Fal-Tr*:* $T \nsubseteq \mathscr{S}_{\mathsf{ID}}^{\boldsymbol{u}}$
  $\Pr\text{-Fal-Tr}_{\mathscr{A},\epsilon,p}(\lambda) = \Pr[\text{Fal-Tr}]$

The EI-TIBIPFE is said to satisfy secure tracing if for any PPT adversary $\mathscr{A}$, polynomial $q(\lambda)$ and non-negligible function $\epsilon(\cdot)$, there exists negligible functions $\mathsf{negl}_1, \mathsf{negl}_2$ satisfying $\epsilon(\lambda) > 1/q(\lambda)$ with the following conditions,

$$\Pr\text{-Fal-Tr}_{\mathscr{A},\epsilon,p}(\lambda) \leq \mathsf{negl}_1, \quad \Pr\text{-Cor-Tr}_{\mathscr{A},\epsilon,p}(\lambda) \geq \Pr\text{-G-D}_{\mathscr{A},\epsilon,p}(\lambda) - \mathsf{negl}_2$$

Note that the notion of EI-TIPFE is a particular case of EI-TIBIPFE where we simply ignore gids used in the syntax of EI-TIBIPFE.

3.9 Embedded Identity Private Linear Identity-Based Inner Product Functional Encryption

An EIPL-IBIPFE for a message vector space $\mathscr{Y} = \{\mathscr{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, user identity space $\mathscr{ID} = \{\{0,1\}^k : k \in \mathbb{N}\}$ and a group identity space $\mathscr{GID} = \{\{0,1\}^{k'} : k' \in \mathbb{N}\}$ consists of five PPT algorithms EIPL-IBIPFE = (Setup, KeyGen, Enc, SplEnc, Dec) and details about these algorithms are given below.

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m) \to$ (msk, mpk, key): The trusted authority takes as input the security parameter λ, the index space n, the 'user identity space' parameter k, the 'group identity space' parameter k', a vector length parameter m, and outputs a master secret key msk, a master public key mpk and a key key. The master public key mpk and the key key are made public while the master secret key msk is kept secret to the trusted authority.

KeyGen(msk, i, id, gid, $\boldsymbol{u}$) $\to \mathsf{sk}_{\boldsymbol{u}}$: On input the master secret key msk, an index $i \in [n]$, an user identity $\mathsf{id} \in \{0,1\}^k$, a group identity $\mathsf{gid} \in \{0,1\}^{k'}$ and a vector $\boldsymbol{u} \in \mathbb{Z}^m$, the trusted authority outputs a secret key $\mathsf{sk}_{\boldsymbol{u}}$.

Enc(mpk, gid$'$, $\boldsymbol{v}$) $\to \mathsf{ct}_{\boldsymbol{v}}$: This algorithm is run by an encryptor by taking input as mpk, a group identity gid$'$, a message vector $\boldsymbol{v} \in \mathbb{Z}^m$ and generates a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ associated to the vector $\boldsymbol{v}$.

SplEnc(key, gid$'$, $\boldsymbol{v}$, (i, ℓ, b)) $\to \mathsf{ct}_{\boldsymbol{v}}$: It outputs a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ by taking input as key, a group identity gid$'$, a message vector $\boldsymbol{v} \in \mathbb{Z}^m$ and index-position-bit tuple $(i, \ell, b) \in [n+1] \times ([k] \cup \{\perp\}) \times \{0,1\}$. If key = mpk then EIPL-IBIPFE is called public key EIPL-IBIPFE, else called private key EIPL-IBIPFE.

Dec$(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) \to \zeta/\perp$: On input the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$, secret key $\mathsf{sk}_{\boldsymbol{u}}$ decryptor outputs the message ζ or a symbol $\perp$ indicating failure.

Correctness. An EIPL-IBIPFE = (Setup, KeyGen, Enc, SplEnc, Dec) scheme is said to be correct if there exists *negligible* functions negl_1, negl_2 such that for all $\lambda, n, k, k', m \in \mathbb{N}$, $\boldsymbol{v} \in \mathbb{Z}^m$, $i \in [n+1]$, $j \in [n]$, user identity $\mathsf{id} \in \{0,1\}^k$, group identity $\mathsf{gid} \in \{0,1\}^{k'}$, $\ell \in ([k] \cup \{\perp\})$, $b \in \{0,1\}$, the following holds,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v}\rangle : \begin{array}{c}(\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m) \\ \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, j, \mathsf{id}, \mathsf{gid}, \boldsymbol{u}) \\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{gid}, \boldsymbol{v})\end{array}\right] \geq 1 - \mathsf{negl}_1(\lambda),$$

If $(j \geq i+1) \vee (i, \ell) = (j, \perp) \vee (i, \mathsf{id}_\ell) = (j, 1-b)$ then the following holds,

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v}\rangle : \begin{array}{c}(\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m) \\ \mathsf{sk}_{\boldsymbol{u}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, j, \mathsf{id}, \mathsf{gid}, \boldsymbol{u}) \\ \mathsf{ct}_{\boldsymbol{v}} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}, \boldsymbol{v}, (i, \ell, b))\end{array}\right] \geq 1 - \mathsf{negl}_2(\lambda).$$

3.10 IND-CPA security of EIPL-IBIPFE

Let $q(\cdot)$ be a fixed polynomial. The security definitions for EIPL-IBIPFE is a generalization from the q-query security notions of EIPL-BE [GKW19] as given below.

Definition 13 (q-query Normal-Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r}(1^n,1^k,1^{k'},1^m) \leftarrow \mathscr{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda,n,1^k,1^{k'},1^m)\\ (\mathsf{gid}^*,\boldsymbol{v}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk})\\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{Enc}(\mathsf{mpk},\mathsf{gid}^*,\boldsymbol{v})\\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\mathsf{gid}^*,\boldsymbol{v},(1,\perp,0))\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IBIPFE scheme is said to satisfy q-query normal-hiding security:

- SplEnc *Oracle: $\mathscr{A}$ can make at most $q(\lambda)$ queries of the form* $(\mathsf{gid}^*,\boldsymbol{v},(1,\ell,\gamma))$.
- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of distinct indices. That is, if $\mathscr{A}$ makes the queries* $(i_1,\mathsf{id}_1,\mathsf{gid}_1,\boldsymbol{u}_1)$, $(i_2,\mathsf{id}_2,\mathsf{gid}_2,\boldsymbol{u}_2)$, …, $(i_\kappa,\mathsf{id}_\kappa,\mathsf{gid}_\kappa,\boldsymbol{u}_\kappa)$, *then $i_a \neq i_b$ when $a \neq b$ for every $a,b \in [\kappa]$.*

Definition 14 (q-query Index-Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r}(1^n,1^k,1^{k'},1^m,i^*) \leftarrow \mathscr{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda,n,1^k,1^{k'},1^m)\\ (\mathsf{gid}^*,\boldsymbol{v}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk})\\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\mathsf{gid}^*,\boldsymbol{v},(i^*,\perp,0))\\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\mathsf{gid}^*,\boldsymbol{v},(i^*+1,\perp,0))\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IBIPFE scheme is said to satisfy q-query index-hiding security:

- SplEnc *Oracle: $\mathscr{A}$ can make at most $q(\lambda)$ queries of the form* $(\mathsf{gid}^*,\boldsymbol{v},(i,\ell,\gamma))$, *where the index i must be equal to either i^* or i^*+1.*
- KeyGen *Oracle: All queries of $\mathscr{A}$ should be of distinct indices and should not be of the form* $(i^*,\mathsf{id},\mathsf{gid}^*,\boldsymbol{u})$. *That is, if $\mathscr{A}$ makes the key queries* $(i_1,\mathsf{id}_1,\mathsf{gid}_1,\ u_1),(i_2,\mathsf{id}_2,\mathsf{gid}_2,\boldsymbol{u}_2)$, …, $(i_\kappa,\mathsf{id}_\kappa,\mathsf{gid}_\kappa,\boldsymbol{u}_\kappa)$, *then $i_a \neq i_b$ when $a \neq b$ for every $a,b \in [\kappa]$ and $i_a \neq i^*$ when* $\mathsf{gid}_a = \mathsf{gid}^*$ *for every $a \in [\kappa]$.*

Definition 15 (q-query Lower Identity-Hiding Security) *If for every stateful PPT adversary $\mathscr{A}$, there exists a negligible function* negl *such that for every $\lambda \in \mathbb{N}$ the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r}(1^n,1^k,1^{k'},1^m,i^*,\ell^*,b^*) \leftarrow \mathscr{A}(1^\lambda)\\ (\mathsf{msk},\mathsf{mpk},\mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda,n,1^k,1^{k'},1^m)\\ (\mathsf{gid}^*,\boldsymbol{v}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk})\\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\mathsf{gid}^*,\boldsymbol{v},(i^*,\perp,0))\\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key},\mathsf{gid}^*,\boldsymbol{v},(i^*,\ell^*,b^*))\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IBIPFE scheme is said to satisfy q-query lower-identity hiding security:

– SplEnc *Oracle:* $\mathscr{A}$ *can make at most* $q(\lambda)$ *queries of the form* $(\mathsf{gid}^*, \boldsymbol{v}, (i, \ell, \gamma))$*, where the index* i *must be equal to* i^**.*
– KeyGen *Oracle: All queries of* $\mathscr{A}$ *should be of distinct indices and should not be of the form* $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ *such that* $\mathsf{id}_{\ell^*} = b^*$*. That is, if* $\mathscr{A}$ *makes the key queries* $(i_1, \mathsf{id}_1, \mathsf{gid}_1, \boldsymbol{u}_1)$*,* $(i_2, \mathsf{id}_2, \mathsf{gid}_2, \boldsymbol{u}_2), \ldots, (i_\kappa, \mathsf{id}_\kappa, \mathsf{gid}_\kappa, \boldsymbol{u}_\kappa)$*, then* $i_a \neq i_b$ *when* $a \neq b$ *for every* $a, b \in [\kappa]$ *and* $i_a \neq i^*$ *or* $(\mathsf{id}_a)_{\ell^*} \neq b^*$ *when* $\mathsf{gid}_a = \mathsf{gid}^*$ *for every* $a \in [\kappa]$*.*

Definition 16 (q-query Upper Identity-Hiding Security) *If for every stateful PPT adversary* $\mathscr{A}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$ *the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}) = \mathfrak{b} : \begin{array}{r}(1^n, 1^k, 1^{k'}, 1^m, i^*, \ell^*, b^*) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m) \\ (\mathsf{gid}^*, \boldsymbol{v}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}}^{(0)} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*+1, \perp, 0)) \\ \mathsf{ct}_{\boldsymbol{v}}^{(1)} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*))\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IBIPFE scheme is said to satisfy q*-query upper identity-hiding security:*

– SplEnc *Oracle:* $\mathscr{A}$ *can make at most* $q(\lambda)$ *queries of the form* $(\mathsf{gid}^*, \boldsymbol{v}, (i, \ell, \gamma))$*, where the index* i *must be equal to either* i^* *or* $i^* + 1$*.*
– KeyGen *Oracle: All queries of* $\mathscr{A}$ *should be of distinct indices and should not be of the form* $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ *such that* $\mathsf{id}_{\ell^*} = 1 - b^*$*. That is, if* $\mathscr{A}$ *makes the key queries* $(i_1, \mathsf{id}_1, \mathsf{gid}_1, \boldsymbol{u}_1)$*,* $(i_2, \mathsf{id}_2, \mathsf{gid}_2, \boldsymbol{u}_2), \ldots, (i_\kappa, \mathsf{id}_\kappa, \mathsf{gid}_\kappa, \boldsymbol{u}_\kappa)$*, then* $i_a \neq i_b$ *when* $a \neq b$ *for every* $a, b \in [\kappa]$ *and* $i_a \neq i^*$ *or* $(\mathsf{id}_a)_{\ell^*} \neq 1 - b^*$ *when* $\mathsf{gid}_a = \mathsf{gid}^*$ *for every* $a \in [\kappa]$*.*

Definition 17 (q-query Message-Hiding Security) *If for every stateful PPT adversary* $\mathscr{A}$*, there exists a negligible function* negl *such that for every* $\lambda \in \mathbb{N}$ *the following holds,*

$$\Pr\left[\mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{r}(1^n, 1^k, 1^{k'}, 1^m, i^*) \leftarrow \mathscr{A}(1^\lambda) \\ (\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow \mathsf{Setup}(1^\lambda, n, 1^k, 1^{k'}, 1^m) \\ (\mathsf{gid}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}) \leftarrow \mathscr{A}^{\mathsf{KeyGen}(\mathsf{msk},\cdot,\cdot,\cdot,\cdot),\mathsf{SplEnc}(\mathsf{key},\cdot,\cdot,\cdot)}(\mathsf{mpk}) \\ \mathfrak{b} \leftarrow \{0,1\}; \mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(\mathfrak{b})}, (i^*, \perp, 0))\end{array}\right] \leq \frac{1}{2} + \mathsf{negl}(\lambda),$$

with the following oracle restrictions then an EIPL-IBIPFE scheme is said to satisfy q*-query message-hiding security:*

– SplEnc *Oracle:* $\mathscr{A}$ *can make at most* $q(\lambda)$ *queries of the form* $(\mathsf{gid}^*, \boldsymbol{v}, (i, \ell, \gamma))$*, where the index* i *must be equal to* i^**.*
– KeyGen *Oracle: All queries of* $\mathscr{A}$ *should be of distinct indices and the form of* $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ *for* $i \geq i^*$ *satisfying the condition* $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$ *if* $\mathsf{gid} = \mathsf{gid}^*$*. That is, if* $\mathscr{A}$ *makes the key queries* $(i_1, \mathsf{id}_1, \mathsf{gid}_1, \boldsymbol{u}_1)$*,* $(i_2, \mathsf{id}_2, \mathsf{gid}_2, \boldsymbol{u}_2), \ldots, (i_\kappa, \mathsf{id}_\kappa, \mathsf{gid}_\kappa, \boldsymbol{u}_\kappa)$*, then* $i_a \neq i_b$ *when* $a \neq b$ *for every* $a, b \in [\kappa]$*, and if* $i_a \geq i^*$ *and* $\mathsf{gid}_a = \mathsf{gid}^*$ *then* $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$ *for any* $a \in [\kappa]$*.*

Note that, the above security notions are described as the Adp-IND-CPA security model. In case of Sel-IND-CPA security model, $\mathscr{A}$ is restricted to submit gid^* before setup for Definitions 13 - 15 whereas $\mathscr{A}$ also submits the message vectors pair $(\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$ along with gid^* before seeing any public parameters. We can similarly define EIPL-IPFE (see Definition 3.6) and its security notions by ignoring the group identities from the above definitions.

4 EI-TIBIPFE from EIPL-IBIPFE

Consider an EIPL-IBIPFE scheme EIPL-IBIPFE = (EIPL-IBIPFE.Setup, EIPL-IBIPFE. KeyGen, EIPL-IBIPFE.Enc, EIPL-IBIPFE.SplEnc, EIPL-IBIPFE.Dec) for a message space $\mathscr{Y} = \{\mathscr{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, a predicate space $\mathscr{X} = \{\mathscr{X}_\lambda\}_{\lambda\in\mathbb{N}}$, a user identity space $\mathscr{ID} = \{0,1\}^k$ and a group identity space $\mathscr{GID} = \{0,1\}^{k'}$. In the following, we provide our EI-TIBIPFE scheme with the same message vector space, user identity space, and group identity space. Depending on the special encryption algorithm of the underlying EIPL-IBIPFE scheme, this generic construction of our EI-TIBIPFE is called *public EI-TIBIPFE* or *private EI-TIBIPFE*.

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m)$ → (msk, mpk, key): The algorithm runs EIPL-IBIPFE setup algorithm as $(\mathsf{msk}, \mathsf{mpk}, \mathsf{key}) \leftarrow$ EIPL-IBIPFE.Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m)$. Outputs master secret key msk, master public key mpk and tracing key key.

KeyGen(msk, i, id, gid, $\boldsymbol{u}$) → $\mathsf{sk}_{\boldsymbol{u}}$: The trusted authority runs the EIPL-IBIPFE key generation algorithm, and generates the secret key as $\mathsf{sk}_{\boldsymbol{u}} \leftarrow$ EIPL-IBIPFE.Key Gen(msk, i, id, gid, $\boldsymbol{u}$).

Enc(mpk, gid′, $\boldsymbol{v}$) → $\mathsf{ct}_{\boldsymbol{v}}$: An encryptor runs the EIPL-IBIPFE encryption algorithm, and outputs the ciphertext as $\mathsf{ct}_{\boldsymbol{v}} \leftarrow$ EIPL-IBIPFE.Enc(mpk, gid′, $\boldsymbol{v}$).

Dec($\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}$) → ζ/ $\perp$: The decrptor runs the EIPL-IBIPFE decryption algorithm, and outputs $\zeta \leftarrow$ EIPL-IBIPFE.Dec($\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}$) or $\perp$ indicating the failure.

Trace$^{\mathscr{D}_{\boldsymbol{u}}}$(key, $1^{\frac{1}{\epsilon(\lambda)}}$, gid, $\boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$) → $\boldsymbol{T}$: Consider two algorithms Index-Trace and ID-Trace defined in Fig. 2 and Fig. 3. First, the Index-Trace algorithm runs for each index $i \in [n]$ and find a collection of indices set T^{index} such that the Index-Trace algorithm outputs 1 corresponds to these indices. Next, the ID-Trace algorithm runs on the index set T^{index} and uses the decoder box to find the required identity of the particular indexed user. Next the tracing algorithm runs ID-Trace algorithm for all indices $i \in T^{\mathsf{index}}$, and for each index i where the ID-Trace algorithm does not output $\perp$. The tracing algorithm adds the output of ID-Trace algorithm to the *identity-set* of traitors T.

1. Set $T^{\mathsf{index}} := \emptyset$. For $i = 1$ to n.
   - Compute $(b, p, q) \leftarrow$ Index-Trace(key, $1^{\frac{1}{\epsilon(\lambda)}}$, gid, $\boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, i$).
   - If $b = 1$, set $T^{\mathsf{index}} := T^{\mathsf{index}} \cup \{(i, p, q)\}$.
2. Set $T^{\mathsf{index}} := \emptyset$. For $\{(i, p, q)\} \in T^{\mathsf{index}}$.
   - Compute id $\leftarrow$ ID-Trace(key, $1^{\frac{1}{\epsilon(\lambda)}}$, gid, $\boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, i$).
   - Set $T := T \cup$ id.
3. Return T.

Correctness. This follows from the correctness of the underlying EIPL-IBIPFE = (Setup, KeyGen, Enc, SplEnc, Dec) scheme. If gid = gid′, the decryptor correctly decrypts the ciphertext using a legitimate secret key; otherwise, it returns $\perp$.

Algorithm: Index-Trace$(\mathsf{key}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{gid}, \boldsymbol{u}, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}, i)$

Inputs: Key key, parameter $\frac{1}{\epsilon(\lambda)}$, a group identity gid, a vector $\boldsymbol{u}$, messages $\boldsymbol{v}^{(0)}$, $\boldsymbol{v}^{(1)}$, index i.
Output: 0/1
It sets $N = \lambda \cdot n/\epsilon$, $\mathsf{count}_1 = \mathsf{count}_2 = 0$. For $j = 1$ to N, it computes the following:

- It chooses $\mathfrak{b}_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}^1_{\boldsymbol{v}^{(\mathfrak{b}_j)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.SplEnc}(\mathsf{key}, \mathsf{gid}, \boldsymbol{v}^{(\mathfrak{b}_j)}, (i, \perp, 0))$ and sends $\mathsf{ct}^1_{\boldsymbol{v}^{(\mathfrak{b}_j)}}$ to $\mathscr{D}_{\boldsymbol{u}}$. Let $\mathfrak{b}'_j \leftarrow \mathscr{D}_{\boldsymbol{u}}(\mathsf{ct}^1_{\boldsymbol{v}^{(\mathfrak{b}_j)}})$.
- If $\mathfrak{b}'_j = \mathfrak{b}_j$ then $\mathsf{count}_1 = \mathsf{count}_1 + 1$, else $\mathsf{count}_1 = \mathsf{count}_1 - 1$.
- Computes $\mathsf{ct}^2_{\boldsymbol{v}^{(\mathfrak{b}_j)}} \leftarrow \mathsf{EIPL\text{-}IPFE.SplEnc}(\mathsf{key}, \mathsf{gid}, \boldsymbol{v}^{(\mathfrak{b}_j)}, (i+1, \perp, 0))$ and sends $\mathsf{ct}^2_{\boldsymbol{v}^{(\mathfrak{b}_j)}}$ to $\mathscr{D}_{\boldsymbol{u}}$. Let $\mathfrak{b}'_j \leftarrow \mathscr{D}_{\boldsymbol{u}}(\mathsf{ct}^2_{\boldsymbol{v}^{(\mathfrak{b}_j)}})$.
- If $\mathfrak{b}'_j = \mathfrak{b}_j$ then $\mathsf{count}_2 = \mathsf{count}_2 + 1$, else $\mathsf{count}_2 = \mathsf{count}_2 - 1$.

If $\frac{\mathsf{count}_1 - \mathsf{count}_2}{N} > \frac{\epsilon}{4n}$, output $\left(1, \frac{\mathsf{count}_1}{N}, \frac{\mathsf{count}_2}{N}\right)$, else output $(0, \perp, \perp)$.

Fig. 2: Index-Trace

Algorithm: ID-Trace$(\mathsf{key}, 1^{\frac{1}{\epsilon(\lambda)}}, \mathsf{gid}, \boldsymbol{u}, \boldsymbol{v}_0, \boldsymbol{v}_1, (i, p, q))$

Inputs: Key key, a group identity gid, parameter $\frac{1}{\epsilon(\lambda)}$, a vector $\boldsymbol{u}$, message pair $\boldsymbol{v}^{(0)}$, $\boldsymbol{v}^{(1)}$, index i, probabilities p, q.
Output: $\mathsf{id} \in \{0,1\}^k$
It sets $N = \lambda \cdot n/\epsilon$, and $\mathsf{count}_\ell = 0$ for $\ell \in [k]$. For $\ell = 1$ to k, it proceeds as follows:

1. For $j = 1$ to N, it computes the following:
   - It chooses $\mathfrak{b}_j \leftarrow \{0,1\}$ and computes $\mathsf{ct}^1_{\boldsymbol{v}^{(\mathfrak{b}_j)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.SplEnc}(\mathsf{key}, \mathsf{gid}, \boldsymbol{v}^{(\mathfrak{b}_j)}, (i, \ell, 0))$ and sends $\mathsf{ct}^1_{\boldsymbol{v}^{(\mathfrak{b}_j)}}$ to $\mathscr{D}_{\boldsymbol{u}}$. Let $\mathfrak{b}'_j \leftarrow \mathscr{D}_{\boldsymbol{u}}(\mathsf{ct}^1_{\boldsymbol{v}^{(\mathfrak{b}_j)}})$.
   - If $\mathfrak{b}'_j = \mathfrak{b}_j$ then $\mathsf{count}_\ell = \mathsf{count}_\ell + 1$, else $\mathsf{count}_\ell = \mathsf{count}_\ell - 1$.

Next, let id be an empty string. For $\ell = 1$ to k, do the following:

1. If $\frac{p+q}{2} > \frac{\mathsf{count}_\ell}{N}$, set $\mathsf{id}_\ell = 0$. Else set $\mathsf{id}_\ell = 1$.

Finally, output id.

Fig. 3: ID-Trace

4.1 Security Analysis

In this subsection, we provide a detailed proof of our EI-TIBIPFE scheme as discussed above.

Theorem 2 *If the our EIPL-IBIPFE scheme is* 1*-query secure as per Definitions 13 to 17, then the above EI-TIBIPFE scheme is secure as per Definitions 11 and 12.*

Proof. We prove this theorem by combining following Theorems 3 and 4.

Theorem 3 (Security of indistinguishability) *If our EIPL-IBIPFE scheme be a* 0*-query secure EIPL-IBIPFE scheme as per Definitions 13 to 17, then our public/private EI-TIBIPFE scheme is secure as per Definition 11.*

Proof. We would like to point out that the scheme EI-TIBIPFE is IND-CPA secure even if the EIPL-IBIPFE scheme satisfies only q-query security. Let us assume that τ be the least integer indexed queried by the adversary of TIBIPFE to the KeyGen oracle. As per Definition 11, all other queried keys by $\mathscr{A}$ to the oracle is of the form $(j, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ where $j \geq \tau$ and satisfying the relation $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$. Also it note that, the adversary $\mathscr{A}$ can not query for a secret key corresponding to the index position less then τ. A high level proof sketch is given below.

We construct a sequence of $2\tau+3$ hybrids experiments to prove IND-CPA security. The sequence of hybrids starts with the Hybrid H_0, which is exactly the same as IND-CPA game.

Hybrid H_0: In this hybrid, the adversary first selectively chooses the challenge group identity gid^* before the challenger sends the master public key mpk to the adversary $\mathscr{A}$. Next, the adversary sends a challenge message vectors $\boldsymbol{v}^{(0)}$, $\boldsymbol{v}^{(1)}$ from to the challenger. The challenger $\mathscr{B}$ generates the challenge ciphertext as $\mathsf{ct}_{\boldsymbol{v}^{(0)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}^{(0)})$ and sends it to $\mathscr{A}$.

Hybrid $H_{i,0}$(for $i \in [\tau]$): This hybrid is identical to the previous hybrid, except that the challenge ciphertext is a special encryption to the index-position-bit tuple $(i, \perp, 0)$ associated with the vector $\boldsymbol{v}^{(0)}$. i.e., $\mathsf{ct}_{\boldsymbol{v}^{(0)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.SplEnc}\ (\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, (i, \perp, 0))$.

Hybrid H_1: This experiment is identical to the hybrid $H_{\tau,0}$ except the challenge message vector is $\boldsymbol{v}^{(1)}$ instead of the vector $\boldsymbol{v}^{(0)}$, i,e., the challenge ciphertext is generated as $\mathsf{ct}_{\boldsymbol{v}^{(1)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(1)}, (\tau, \perp, 0))$.

Hybrid $H_{\tau-i+1,1}$(for $i \in [\tau]$): This experiment is identical to the previous hybrid H_1 except that the adversary gets the challenge ciphertext $\mathsf{ct}_{\boldsymbol{v}^{(1)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(1)}, (\tau - i + 1, \perp, 0))$ corresponding to the tuple $(\tau - i + 1, \perp, 0)$.

Hybrid H_2: This hybrid is similar to hybrid $H_{1,1}$ expect the challenge ciphertext is of the form $\mathsf{ct}_{\boldsymbol{v}^{(1)}} \leftarrow \mathsf{EIPL\text{-}IBIPFE.Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}^{(1)})$.
For any PPT adversary $\mathscr{A}$, let $p_{\mathscr{A},x}()$ be a function of λ that denotes the probability of $\mathscr{A}$ outputting 0 in Hybrid H_x. Note that $p_{\mathscr{A},0} - p_{\mathscr{A},2}$ is the advantage of $\mathscr{A}$ in the IND-CPA security game.

Claim 1. If the scheme EIPL-IBIPFE is 0-query normal-hiding secure as per Definition 13, then for any PPT adversary $\mathscr{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|p_{\mathscr{A},0} - p_{\mathscr{A},1,0}| \leq \mathsf{negl}(\lambda)$.

Proof of Claim 1. Since, the adversary can not query for the secret key corresponding to the index position $i < \tau$. Since we assume that it holds the normal-hiding security, so the adversary can not learn anything about the challenge bit $\mathfrak{b}'$ from the $\mathsf{EIPL\text{-}IBIPFE.Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}^{(0)})$ and $\mathsf{EIPL\text{-}IBIPFE.Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, (1, \perp, 0))$ with non-negligible advantages.

Claim 2. If the scheme EIPL-IBIPFE is 0-query index-hiding secure as per Definition 14, then for any PPT adversary $\mathscr{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and $i \in [\tau]$, $|p_{\mathscr{A},i-1,0} - p_{\mathscr{A},i,0}| \leq \mathsf{negl}(\lambda)$.

Proof of Claim 2. Since all the key queries are made by the adversary to the key generation oracle corresponds to the index greater then or equal to τ. So the adversary can not decrypt the special encryption of $\mathsf{EIPL\text{-}IBIPFE.SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, (i, \perp, 0))$ for $i \in [\tau]$. The above probabilities difference directly follows from q-query normal-hiding security of EIPL-IBIPFE.

Claim 3. If the scheme EIPL-IBIPFE is 0-query message-hiding secure as per Definition 17, then for any PPT adversary $\mathscr{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|p_{\mathscr{A},\tau,0} - p_{\mathscr{A},\tau,1}| \leq \mathsf{negl}(\lambda)$.

Proof of Claim 3. Although, adversary gets the secret key for the index τ, but from the message-hiding security $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$. Therefore, the adversary can not extract the challenge bit $\mathfrak{b}$ from the inner product values. So, the above probabilities difference directly follows from the q-query message-hiding security of EIPL-IBIPFE.

Claim 4. If the scheme EIPL-IBIPFE is 0-query index-hiding secure as per Definition 14, then for any PPT adversary $\mathscr{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ and $i \in [\tau]$, $|p_{\mathscr{A},\tau-i+1,1} - p_{\mathscr{A},\tau-i,1}| \leq \mathsf{negl}(\lambda)$.

Directly follows from index-hiding security as Claim 2.

Claim 5. If the scheme EIPL-IBIPFE is 0-query normal-hiding secure as per Definition 6, then for any PPT adversary $\mathscr{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $|p_{\mathscr{A},1} - p_{\mathscr{A},1,1}| \leq \mathsf{negl}(\lambda)$.

Directly follows from index-hiding security as Claim 5.

Combining the above claims, we get proof of Theorem. 3. □

Theorem 4 (Security of Tracing) *Let us consider an EIPL-IBIPFE scheme is 1-query secure as per the Definitions 13 to Definition 17, then our traceable identity-based inner product functional encryption scheme TIBIPFE is secure as per the Definitions 12.*

Proof of the tracing is similar to the [GKW19], but for completeness we include this proof into the Appendix A.

5 EIPL-IBIPFE from IBIPFE

Consider IBIPFE = (IBIPFE.Setup, IBIPFE.KeyGen, IBIPFE.Enc, IBIPFE.Dec) be an identity-based inner product functional encryption scheme with the predicate space $\mathscr{X} = \{\mathscr{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and a message space $\mathscr{Y} = \{\mathscr{Y}_\lambda\}_{\lambda\in\mathbb{N}}$. In the following, we discuss the generic construction of EIPL-IBIPFE from IBIPFE.

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m) \to$ (msk, mpk, key): On input the security parameter λ, identity parameter k, length of the vector m and index space n, the trusted authority performs the following steps:

- run the IBIPFE setup algorithm $2nk$ times and generates $\{\mathsf{IBIPFE.msk}_{i,\ell,b}, \mathsf{IBIPFE.mpk}_{i,\ell,b}\} \leftarrow \mathsf{IBIPFE.Setup}(1^\lambda, 1^m, 1^{k'})$ for all $(i,\ell,b) \in [n] \times [k] \times \{0,1\}$.
- set the master secret key $\mathsf{msk} = \{\mathsf{IBIPFE.msk}_{i,\ell,b}\}_{(i,\ell,b)\in[n]\times[k]\times\{0,1\}}$ and the master public key $\mathsf{mpk} = \{\mathsf{IBIPFE.mpk}_{i,\ell,b}\}_{(i,\ell,b)\in[n]\times[k]\times\{0,1\}}$ with the key $\mathsf{key} = \mathsf{mpk}$.

KeyGen(msk, i, id, gid, $\boldsymbol{u}$) $\to \mathsf{sk}_{\boldsymbol{u}}$: On input the master secret key msk, an index $i \in [n]$, an user identity $\mathsf{id} \in \{0,1\}^k$, a group identity $\mathsf{gid} \in \{0,1\}^{k'}$, and a vector $\boldsymbol{u} \in \mathbb{Z}^m$, the algorithm executes the following steps:

- run IBIPFE key generation algorithm k times, and then it generates $\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell} \leftarrow \mathsf{IBIPFE.KeyGen}\,(\mathsf{IBIPFE.msk}_{i,\ell,\mathsf{id}_\ell}, \mathsf{gid}, \boldsymbol{u})$ for all $\ell \in [k]$.
- output the secret key $\mathsf{sk}_{\boldsymbol{u}} = \big(\mathsf{id}, \mathsf{gid}, \{\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell}\}_{\ell\in[k]}\big)$.

Enc(mpk, gid$'$, $\boldsymbol{v}$) $\to \mathsf{ct}_{\boldsymbol{v}}$: To encrypt the vector $\boldsymbol{v} \in \mathbb{Z}^m$, the encryptor performs the following mechanism:

- randomly choose $\boldsymbol{v}_{i,\ell} \leftarrow \mathbb{Z}^m$ for $i \in [n]$, $\ell \in [k-1]$ and set $\boldsymbol{v}_{i,k} = \boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell} \;\; \forall\; i \in [n]$.
- run the IBIPFE encryption algorithm $2nk$ times and generate $\mathsf{IBIPFE.ct}_{i,\ell,b} \leftarrow \mathsf{IBIPFE.Enc}\,(\mathsf{IBIPFE.mpk}_{i,\ell,b}, \mathsf{gid}', \boldsymbol{v}_{i,\ell})$ for all $(i,\ell,b) \in ([n] \times [k] \times \{0,1\})$.
- output the ciphertext $\mathsf{ct}_{\boldsymbol{v}} = \{\mathsf{IBIPFE.ct}_{i,\ell,b}\}_{(i,\ell,b)\in([n]\times[k]\times\{0,1\})}$.

$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}', \boldsymbol{v}, (i^*, \ell^*, b^*)) \to \mathsf{ct}_{\boldsymbol{v}}$: The encryption algorithm first parses the key key be as defined during setup and possess the following steps:

- choose $n \cdot (k-1)$ uniformly random vectors as $\boldsymbol{v}_{i,\ell} \leftarrow \mathbb{Z}^m$ for $i \in [n]$, $\ell \in [k-1]$, Next, set $\boldsymbol{v}_{i,k}$ as

$$\boldsymbol{v}_{i,k} = \begin{cases} \boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell} & \text{if } i \geq i^* \\ \leftarrow \mathbb{Z}^m & \text{otherwise} \end{cases}$$

  where $\boldsymbol{v}_{i,k} \leftarrow \mathbb{Z}^m$ represents sampling $\boldsymbol{v}_{i,k}$ as a random vector.
- set the vectors $\widetilde{\boldsymbol{v}}_{i,\ell,b}$ as:

$$\widetilde{\boldsymbol{v}}_{i,\ell,b} = \begin{cases} \boldsymbol{v}_{i,\ell} & \text{if } (i,\ell,b) \neq (i^*, \ell^*, b^*) \\ \leftarrow \mathbb{Z}^m & \text{otherwise} \end{cases}$$

- run the IBIPFE encryption algorithm $2nk$ times and generate the ciphertext $\mathsf{ct}_{i,\ell,b} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{IBIPFE.mpk}_{i,\ell,b}, \mathsf{gid}', \widetilde{\boldsymbol{v}}_{i,\ell,b})$ for all $(i,\ell,b) \in ([n] \times [k] \times \{0,1\})$.
- Finally, output $\mathsf{ct}_{\boldsymbol{v}} = \{\mathsf{ct}_{i,\ell,b}\}_{(i,\ell,b)\in([n]\times[k]\times\{0,1\})}$.

$\mathsf{Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) \to \zeta / \perp$: To decrypt the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$, this algorithm performs as follows:

- run the IBIPFE decryption algorithm and compute

$$z_\ell \leftarrow \mathsf{IBIPFE.Dec}(\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell}, \mathsf{IBIPFE.ct}_{i,\ell,\mathsf{id}_\ell}) \text{ for all } \ell \in [k].$$

- output $\zeta = \sum_{\ell\in[k]} z_\ell$.

5.1 Correctness.

If $\mathsf{gid} = \mathsf{gid}'$ then it proceeds as below, otherwise returns $\perp$. Note that, for all $i \in [n]$, $\sum_{\ell=1}^{k} \boldsymbol{v}_{i,\ell} = \boldsymbol{v}$ in normal encryption. Therefore,

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{\ell=1}^{k} \langle \boldsymbol{u}, \boldsymbol{v}_{i,\ell} \rangle = \sum_{\ell=1}^{k} \mathsf{IBIPFE.Dec}(\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell}, \mathsf{IBIPFE.ct}_{i,\ell,\mathsf{id}_\ell}) = \sum_{\ell=1}^{k} z_\ell.$$

In special encryption we note that,

$$(i \geq i^* + 1) \vee (i,\ell) = (i^*, \ell^* = \perp) \vee (i^*, \mathsf{id}_{\ell^*}) = (i, 1-b^*) \implies \widetilde{\boldsymbol{v}}_{i,\ell,b} = \boldsymbol{v}_{i,\ell}.$$

Thus, for $i \geq i^*$, we have $\boldsymbol{v} = \sum_{\ell=1}^{k} \boldsymbol{v}_{i,\ell}$. Therefore,

$$\begin{aligned} \langle \boldsymbol{u}, \boldsymbol{v} \rangle &= \sum_{\ell=1}^{k} \langle \boldsymbol{u}, \widetilde{\boldsymbol{v}}_{i,\ell,b} \rangle \text{ if } (i \geq i^* + 1) \vee (i,\ell) = (i^*, \ell^* = \perp) \vee (i^*, \mathsf{id}_{\ell^*}) = (i, 1-b^*) \\ &= \sum_{\ell=1}^{k} \mathsf{IBIPFE.Dec}(\mathsf{IBIPFE.sk}_{i,\ell,\mathsf{id}_\ell}, \mathsf{IBIPFE.ct}_{i,\ell,\mathsf{id}_\ell}) = \sum_{\ell=1}^{k} z_\ell. \end{aligned}$$

5.2 Security Analysis

Here, we prove the security of our generic EIPL-IBIPFE construction from IBIPFE.

Theorem 5 *If the underlying IBIPFE = (IBIPFE.Setup, IBIPFE.KeyGen, IBIPFE.Enc, IBIPFE.Dec) scheme is IND-CPA secure, then our EIPL-IBIPFE =(Setup, KeyGen, Enc, SplEnc, Dec) scheme is secure as per Definitions 13 to 17.*

Proof. We prove the above theorem by combining of following Lemmas 1 to 5. The proof of each lemma holds directly from the IND-CPA security of underlying IBIPFE scheme. We also note that our construction preserve the security level of the underlying IBIPFE, i.e., the proposed EIPL-IBIPFE is selectively or adaptively secure if the IBIPFE is so. Next, we discuss the formal proof.

Lemma 1 *Assuming the underlying IBIPFE is IND-CPA secure, then our EIPL-IBIPFE scheme satisfies the normal-hiding security as per the Definition 13.*

Proof. The distribution of normal encryption and special encryption ciphertexts are identical since the special encryption algorithm runs on input (i^*, ℓ^*, b^*) i.e., it computes SplEnc(key, $\mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*)$) with $i^* = 1$, $\ell^* = \perp$. Therefore, $\widetilde{\boldsymbol{v}}_{i,\ell,b} = \boldsymbol{v}_{i,\ell}$ for all i, ℓ, b as the condition $(i > i^*) \vee ((i = i^*) \wedge (\ell \neq \ell^* \vee b \neq b^*))$ is equivalent to $i \geq 1$. □

Lemma 2 *Assuming the underlying IBIPFE is IND-CPA secure, then our EIPL-IBIPFE scheme satisfies the index-hiding security as per the Definition 14.*

Proof. We will prove this security of indistinguishability via. IND-CPA security of the underlying IBIPFE scheme. Recall that the index-hiding security requires the special encryption to $(i^*, \perp, 0)$ is indistinguishable from the special encryption to $(i^*+1, \perp, 0)$ if the adversary is not provided with a key of the form $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$. In the table below, we represent the IBIPFE ciphertexts that are different in the special encryptions to $(i^*, \perp, 0)$ and $(i^*+1, \perp, 0)$.

$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \perp, 0))$	$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*+1, \perp, 0))$
$\mathsf{ct}_{i^*,k,0} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,k,0}, \mathsf{gid}^*, \boldsymbol{w})$	$\mathsf{ct}_{i^*,k,0} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,k,0}, \mathsf{gid}^*, \boldsymbol{w})$
$\mathsf{ct}_{i^*,k,1} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,k,1}, \mathsf{gid}^*, \boldsymbol{w})$	$\mathsf{ct}_{i^*,k,1} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,k,1}, \mathsf{gid}^*, \boldsymbol{w})$
where $\boldsymbol{w} = \boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i^*,\ell}$	where $\boldsymbol{w} \leftarrow \mathbb{Z}^m$.

Note that, according to the game restriction, $\mathscr{A}$ can not query to KeyGen oracle for an IBIPFE secret key corresponding to the index-group identity pair (i^*, gid^*). Therefore, $\mathscr{A}$ is not allowed to make any IBIPFE secret key of the form $\mathsf{IBIPFE.sk}_{i^*,\ell,\mathsf{id}_\ell}$ for a vector $\boldsymbol{u}$ which corresponds to the challenge group identity gid^*. Hence, by IND-CPA security of the underlying IBIPFE scheme, we first change the ciphertext $\mathsf{ct}_{i^*,k,0}$ to encrypt a random vector $\boldsymbol{w}$ instead of encrypting $\boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i^*,\ell}$ and then change the ciphertext $\mathsf{ct}_{i^*,k,1}$ to encrypt the same random vector $\boldsymbol{w}$ instead of encrypting $\boldsymbol{v} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i^*,\ell}$. We observe that we can apply the IND-CPA security even if $\langle \boldsymbol{u}, \boldsymbol{v} \rangle \neq \langle \boldsymbol{u}, \boldsymbol{w} \rangle$ as in all the queried secret keys of $\mathscr{A}$ we have $\mathsf{gid} \neq \mathsf{gid}^*$. Hence, the index-hiding security follows from the IND-CPA security of the IBIPFE scheme. □

Lemma 3 *Assuming the underlying IBIPFE is IND-CPA secure, then our EIPL-IBIPFE satisfies the lower identity-hiding security as per the Definition of 15.*

Proof. We recall that in the lower identity-hiding security it is required that the special encryption to $(i^*, \perp, 0)$ is indistinguishable from the special encryption to (i^*, ℓ^*, b^*), given that the adversary does not have a secret key for $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = b^*$. In the table below, we represent the IBIPFE ciphertexts that are different in the special encryptions to

$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \perp, 0))$	$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*))$
$\mathsf{ct}_{i^*,\ell^*,b^*} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,\ell^*,b^*}, \mathsf{gid}^*, \boldsymbol{w})$ where $\boldsymbol{w} = \boldsymbol{v} - \sum_{\ell \neq \ell^*} \boldsymbol{v}_{i^*,\ell}$	$\mathsf{ct}_{i^*,\ell^*,b^*} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,\ell^*,b^*}, \mathsf{gid}^*, \boldsymbol{w})$ where $\boldsymbol{w} \leftarrow \mathbb{Z}^m$

$(i^*, \perp, 0)$ and (i^*, ℓ^*, b^*).
We observe that no IBIPFE secret key is generated using $\mathsf{msk}_{i^*,\ell^*,b^*}$ for a pair $(\mathsf{gid}^*, \boldsymbol{u})$ with $\mathsf{id}_{\ell^*} = b^*$. Therefore, depending on the IND-CPA security of the IBIPFE, we can change the ciphertext component $\mathsf{ct}_{i^*,\ell^*,b^*}$ in $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \perp, 0))$ from encrypting the vector $\boldsymbol{w} = \boldsymbol{v} - \sum_{\ell \neq \ell^*} \boldsymbol{v}_{i^*,\ell}$ to a random vector $\boldsymbol{w} \leftarrow \mathbb{Z}^m$. Hence, the proof follows. □

Lemma 4 *Assuming the underlying IBIPFE is IND-CPA secure, then our EIPL-IBIPFE scheme satisfies the upper identity-hiding security as per the Definition 16.*

Proof. We recall that in the lower identity-hiding security it is required that the special encryption to (i^*, ℓ^*, b^*) is indistinguishable from the special encryption to $(i^* + 1, \perp, 0)$, given the adversary does not have a secret key for $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = 1 - b^*$. In the table below, we represent the IBIPFE ciphertexts that are different in the special encryptions to (i^*, ℓ^*, b^*) and $(i^* + 1, \perp, 0)$.

$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*))$	$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^* + 1, \perp, 0))$
$\mathsf{ct}_{i^*,\ell^*,1-b^*} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,\ell^*,1-b^*}, \mathsf{gid}^*, \boldsymbol{w})$ where $\boldsymbol{w} = \boldsymbol{v} - \sum_{\ell \neq \ell^*} \boldsymbol{v}_{i^*,\ell}$	$\mathsf{ct}_{i^*,\ell^*,1-b^*} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,\ell^*,1-b^*}, \mathsf{gid}^*, \boldsymbol{w})$ where $\boldsymbol{w} \leftarrow \mathbb{Z}^m$

Using a similar argument as above, one can show that the IND-CPA security of IBIPFE ensures the indistinguishability of $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^*, \ell^*, b^*))$ and $\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}, (i^* + 1, \perp, 0))$. □

Lemma 5 *Assuming the underlying IBIPFE is IND-CPA secure, then our EIPL-IBIPFE scheme satisfies the message-hiding security as per the Definition 17.*

Proof. The message-hiding security requires that the special encryptions of $\boldsymbol{v}^{(0)}$ and $\boldsymbol{v}^{(1)}$ to the same tuple $(i^*, \perp, 0)$ under gid^* are indistinguishable if all the secret key queries of the form $(i \geq i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ must satisfy $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$. In the table below, we represent the IBIPFE ciphertexts that are different in the special encryptions of $\boldsymbol{v}^{(0)}$ and $\boldsymbol{v}^{(1)}$.

$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, (i^*, \perp, 0))$	$\mathsf{SplEnc}(\mathsf{key}, \mathsf{gid}^*, \boldsymbol{v}^{(1)}, (i^*, \perp, 0))$
$\mathsf{ct}^{(0)}_{i^*,k,b} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,k,b}, \mathsf{gid}^*, \boldsymbol{v}^{(0)}_{i,k})$ for $b \in \{0,1\}$ where $\boldsymbol{v}^{(0)}_{i,k} = \boldsymbol{v}^{(0)} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell}$ for all $i \geq i^*$	$\mathsf{ct}^{(1)}_{i^*,k,b} \leftarrow \mathsf{IBIPFE.Enc}(\mathsf{mpk}_{i^*,k,b}, \mathsf{gid}^*, \boldsymbol{v}^{(1)}_{i,k})$ for $b \in \{0,1\}$ where $\boldsymbol{v}^{(1)}_{i,k} = \boldsymbol{v}^{(1)} - \sum_{\ell=1}^{k-1} \boldsymbol{v}_{i,\ell}$ for all $i \geq i^*$

Note that, any secret key for the tuple $(i \geq i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ must satisfy $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)}_{i,k} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)}_{i,k} \rangle$. Therefore, $\mathsf{ct}^{(0)}_{i^*,k,b}$ and $\mathsf{ct}^{(1)}_{i^*,k,b}$ are indistinguishable for $b \in \{0,1\}$, $i \geq i^*$, by the IND-CPA security of the IBIPFE. □

Hence, the proof follows. □

6 Adaptively secure EIPL-IPFE using Bilinear Maps

Let us assume $\mathcal{G}_{\mathsf{BG.Gen}}$ be a bilinear group generator of a composite order group $N = p \cdot q$ where p, q be two prime integers. Now, we describe our EIPL-IPFE = (Setup, KeyGen, Enc, SplEnc, Dec) scheme based on bilinear map.

Setup$(1^\lambda, n, 1^k, 1^m) \to$ (msk, mpk, key): A trusted authority takes a security parameter λ, the identity space parameter k, index space parameter n and message vector length m as input and executes the algorithm as follows:

- sets $\widetilde{n} = \lceil \sqrt{\frac{n}{k}} \rceil$ and $\widehat{n} = \lceil \frac{n}{\widetilde{n}} \rceil$.
- samples a bilinear group $\mathbb{BG} = (p, q, N = p \cdot q, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot)) \leftarrow \mathcal{G}_{\mathsf{BG.Gen}}(1^\lambda)$.
- chooses random generators $g_p, h_p, f_p \in \mathbb{G}_p$ and $g_q, h_q, f_q \in \mathbb{G}_q$. It sets $g = g_p g_q, h = h_p h_q, f = f_p f_q \in \mathbb{G}$. Also, samples some random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, &\quad c_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], &\quad r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

- samples $\beta \leftarrow \mathbb{Z}_q$ and sets

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, h, g, f, E_q = g_q^\beta, Z_q = f_q^\beta \\ \left\{ \begin{array}{c} E_{x,j} = g^{r_{x,j}}, F_{x,j} = h^{r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = g_q^{\beta r_{x,j}}, F_{q,x,j} = h_q^{\beta r_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}} \\ W_{q,x,j} = e(f_q, f_q)^{\beta \psi_{x,j}}, W_{x,j} = e(f,f)^{\psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b} = g^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

$$\mathsf{msk} = \left(\begin{array}{c} \mathbb{G}, g, \{r_{x,j}, \alpha_{x,j}, \psi_{x,j}\}_{x \in [\widehat{n}], j \in [m]}, \\ \{c_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}} \end{array} \right), \quad \mathsf{key} = \mathsf{mpk}$$

- Finally, it publishes the master public key mpk and keeps the master secret key msk secret to itself.

KeyGen(msk, i, id, $\boldsymbol{u}$) $\to \mathsf{sk}_{\boldsymbol{u}}$: On input the master secret key msk, an index $i \in [n]$, an identity $\mathsf{id} \in \{0,1\}^k$ and a vector $\boldsymbol{u} \in \mathbb{Z}^m$, the trusted authority proceeds as follows:

- consider $(x, y) \in [\widehat{n}] \times [\widetilde{n}]$ be the unique row wise representation of index i (for any $i \in [n]$, its corresponding indices can be defined as $y = i \mod \widetilde{n}$ and $x = \lceil \frac{i}{\widetilde{n}} \rceil$)
- it generates the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, K = (K_1, K_2))$ where

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle} \; ; \; K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

Enc(mpk, $\boldsymbol{v}$) $\to \mathsf{ct}_{\boldsymbol{v}}$: The encryption algorithm is the same as special encryption algorithm (SplEnc) when run on index-position-bit tuple $(i^*, \ell^*, b^*) = (1, \perp, 0)$.

SplEnc(key, $\boldsymbol{v}$, (i^*, ℓ^*, b^*)) $\rightarrow \mathsf{ct}_{\boldsymbol{v}}$: The encryptor takes input as key, a vector $\boldsymbol{v} \in \mathbb{Z}^m$, index-position-bit tuple (i^*, ℓ^*, b^*) and performs the following steps:

– let $(x^*, y^*) \in [\widehat{n}] \times [\widetilde{n}]$ be the unique row-wise representation of the index i^*.
– chooses random exponents as

$$\forall j \in [m],\ \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N,\ \tau, t \in \mathbb{Z}_N,\ \ \forall x \in [\widehat{n}],\ \ s_x, e_x, \kappa_x, f_x, d_x \leftarrow \mathbb{Z}_N$$
$$\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\},\ \ w_{y,\ell,b}, v_{y,\ell,b} \leftarrow \mathbb{Z}_N$$

– for all $x \in [\widehat{n}]$, $j \in [m]$ and $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$, it generates the following components as described in the Table 2 and Table 3.

Table 2: Computing row components for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t}$	$Z_q^{\kappa_x t}$	$e(gq, gq)^{\nu_j} \cdot G_{q,x,j}^{t s_x} \cdot W_{q,x,j}^{t \kappa_x}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$g^{s_x t}$	$f^{\kappa_x t}$	$e(g,g)^{\nu_j} \cdot G_{x,j}^{t s_x} \cdot W_{x,j}^{t \kappa_x}$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	f^{d_x}	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 3: Computing column components for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b} \tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee (y, \ell, b) = (y^*, \ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b} \tau} \cdot V_{\ell,b}^{v_{y,\ell,b} \tau}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$

– outputs the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ associated with the vector $\boldsymbol{v}$ as

$$\mathsf{ct} = \begin{pmatrix} \{R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}\}_{x \in [\widehat{n}], j \in [m]}, \\ \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}} \end{pmatrix}$$

Dec(sk$_{\boldsymbol{u}}$, ct$_{\boldsymbol{v}}$) $\rightarrow \zeta$/ $\perp$: The decryptor uses a secret key $\mathsf{sk}_{\boldsymbol{u}}$ to decrypt the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$. Then, it computes

$$\eta = \frac{\prod_{j \in [m]} I_{x,j}{}^{u_j} \cdot \prod_{j \in [m]} e(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j})}{\prod_{j \in [m]} e(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}) \cdot e(K_1, A_x) \cdot e(K_2, B_x)}$$

In all group-based IPFE schemes [ALS16, ABCP15], it is assumed that the inner product values lies in a polynomial range. It is also the same for our scheme. Therefore, after recovering η, we compute inner product value ζ through an exhaustive search over the range of the inner product values, i.e., it returns $\zeta = \log \eta$.

6.1 Correctness

Theorem 6 *If all the components are generated as above algorithms, then our propose EIPL-IPFE scheme is correct with non-negligible probability.*

Proof. Consider the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, K)$ corresponding to the index $i = (x, y)$, an identity id and a vector $\boldsymbol{u}$. We know that $K = (K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell})^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle}, K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle})$ and consider a ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ is an encryption of a vector $\boldsymbol{v}$ and index-position-bit tuple

(i^*, ℓ^*, b^*). It consists of $\mathsf{ct}_{\boldsymbol{v}} = (\{R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}\}_{x,j}, \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{y,\ell,b})$. From the definition of EIPL-IPFE, correctness holds or the decryption oracle gives the outputs $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ if $(i \geq i^* + 1) \vee ((i^*, \ell^*) = (i, \perp)) \vee ((i^*, \mathsf{id}_{\ell^*}) = (i, 1 - b^*))$. Consider $i^* = (x^*, y^*)$. Let the index, identity pair (i, id) satisfies the above restrictions. Depending on the representation of i, we consider the following cases:

Case 1: $x > x^*$ In this case, we have all the row components for all $x \in [\widetilde{n}], j \in [m]$ as $R_{x,j} = E_{q,x,j}^{s_x}, \widetilde{R}_{x,j} = F_{q,x,j}^{s_x \tau}, A_x = E_q^{s_x t}, B_x = Z_q^{\kappa_x t}, I_{x,j} = e(g_q, g_q)^{v_j} \cdot G_{q,x,j}^{t s_x} \cdot W_{q,x,j}^{t \kappa_x}$. The decryption does not depend whether $y > y^*$ or not, we can compute the following components from the Table 2 and Table 3. First consider $(y > y^*) \vee ((y = y^*) \wedge (\ell, b) \neq (\ell^*, b^*))$, then we compute following components:

$$\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j \in [m]} e\left(g_q^{\beta r_{x,j} s_x}, \prod_{\ell \in [k]} g^{c_{y,\ell,\mathsf{id}_\ell} t u_j} h^{w_{y,\ell,\mathsf{id}_\ell} \tau u_j}\right)$$
$$= e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot e(g_q, h_q)^{\beta s_x \tau \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} w_{y,\ell,\mathsf{id}_\ell}} \tag{1}$$

$$\prod_{j \in [m]} {I_{x,j}}^{u_j} = e(g_q, g_q)^{\sum_{j \in [m]} u_j v_j} \cdot e(g_q, g_q)^{\sum_{j \in [m]} \beta \alpha_{x,j} t s_x u_j} \cdot e(f_q, f_q)^{\sum_{j \in [m]} \beta \psi_{x,j} t \kappa_x u_j}$$
$$= e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v} \rangle} \cdot e(g_q, g_q)^{\beta t s_x \langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot e(f_q, f_q)^{\beta t \kappa_x \langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \tag{2}$$

$$\prod_{j \in [m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j \in [m]} e\left(h_q^{\beta r_{x,j} s_x \tau}, \prod_{\ell \in [k]} g^{w_{y,\ell,\mathsf{id}_\ell} u_j}\right)$$
$$= e(h_q, g_q)^{\beta s_x \tau \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} w_{y,\ell,\mathsf{id}_\ell}} \tag{3}$$

$$e(K_1, A_x) = e(g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle + \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}}, g_q^{\beta s_x t})$$
$$= e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \tag{4}$$
$$e(K_2, B_x) = e(f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}, f_q^{\beta \kappa_x t}) = e(f_q, f_q)^{\beta \kappa_x t \langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \tag{5}$$

Therefore, we get that for every $\ell \in [k], j \in [m]$,

$$\frac{\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)}{\prod_{j \in [m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)} = e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \tag{6}$$

So, from Equ. 2,4,5,6 we have,

$$\frac{\prod_{j \in [m]} {I_{x,j}}^{u_j} \cdot \prod_{j \in [m]} e(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j})}{\prod_{j \in [m]} e(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}) \cdot e(K_1, A_x) \cdot e(K_2, B_x)}$$
$$= e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot \frac{e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}}{e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}}} = e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}$$

Finally, it returns $\log e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}$.

Now, we consider $(y < y^*) \vee ((y, \ell, b) = (y^*, \ell^*, b^*))$, then we compute the following components.

$$\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)$$

$$= \prod_{j\in[m]} e\left(g_q^{\beta r_{x,j} s_x}, \prod_{\ell\in[k]} g^{c_{y,\ell,\mathsf{id}_\ell} t u_j} h^{w_{y,\ell,\mathsf{id}_\ell} \tau u_j} h^{\tau\delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell} u_j}\right)$$

$$= e(g_q, g_q)^{\beta s_x t\langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot e(g_q, h_q)^{\beta s_x \tau\langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} (w_{y,\ell,\mathsf{id}_\ell} + \delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell})} \tag{7}$$

$$\prod_{j\in[m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)$$

$$= \prod_{j\in[m]} e\left(h_q^{\beta r_{x,j} s_x \tau}, \prod_{\ell\in[k]} g^{w_{y,\ell,\mathsf{id}_\ell} u_j} \cdot g^{\delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell} u_j} \cdot g_p^{\gamma_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell} u_j}\right)$$

$$= e(h_q, g_q)^{\beta s_x \tau\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell} + \delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell}} \tag{8}$$

In this case also,

$$\frac{\prod_{j\in[m]} e\left(R_{x,j}, \prod_{\ell\in[k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)}{\prod_{j\in[m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)} = e(g_q, g_q)^{\beta s_x t\langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}}$$

So, correct decryption follows as previous.

Case 2: (Otherwise) In this case, the correctness needs to hold if $(i^*, \ell^*) = (i, \perp) \vee (i^*, \mathsf{id}_{\ell^*}) = (i^*, 1 - \mathsf{id}_{\ell^*})$ or we can write as $(x = x^*) \wedge ((y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*)))$

$$\prod_{j\in[m]} e\left(R_{x,j}, \prod_{\ell\in[k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j\in[m]} e\left(g^{r_{x,j} s_x}, \prod_{\ell\in[k]} g^{c_{y,\ell,\mathsf{id}_\ell} t u_j} h^{w_{y,\ell,\mathsf{id}_\ell} \tau u_j}\right)$$

$$= e(g, g)^{s_x t\langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot e(g, h)^{s_x \tau\langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell}} \tag{9}$$

$$\prod_{j\in[m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j\in[m]} e\left(h^{r_{x,j} s_x \tau}, \prod_{\ell\in[k]} g^{w_{y,\ell,\mathsf{id}_\ell} u_j}\right)$$

$$= e(h, g)^{s_x \tau\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell}} \tag{10}$$

$$e(K_1, A_x) = e(g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle + \langle \boldsymbol{r}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}}, g^{s_x t})$$

$$= e(g, g)^{s_x t\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot e(g, g)^{s_x t\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} \tag{11}$$

$$e(K_2, B_x) = e(f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}, f^{\kappa_x t})$$

$$= e(f, f)^{\beta\kappa_x t\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \tag{12}$$

$$\prod_{j\in[m]} I_{x,j}{}^{u_j} = e(g, g)^{\langle \boldsymbol{u}, \boldsymbol{v}\rangle} e(g, g)^{t s_x\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} e(f, f)^{t\kappa_x\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \tag{13}$$

By the same computations as case 1, the correctness of EIPL-IPFE holds. □

6.2 Security Analysis

This section shows the adaptive security of our EIPL-IPFE from the bilinear map. To prove the security, let us consider the following Theorem.

Theorem 7 *If the assumptions 3,4,5,6 and 7 hold over the bilinear group $\mathbb{BG}$, then our EIPL-IPFE scheme is secure as per the Definition 6 to 10.*

Proof. To prove the security of our scheme, we show that our construction satisfies all five security properties. We significantly modify the proof technique of [BW06, GKW19] to compatible it with our scheme. Before going the main idea of the proof technique, we would like to focus that since our EIPL-IPFE consists of a public key special encryption algorithm thus, the adversary does not need to query to EIPL-IPFE challenger for the special encryption queries or ciphertext queries. Therefore, the adversary only performs secret key queries to the challenger throughout security game.

Lemma 6 (Normal-hiding) *Our EIPL-IPFE is normal-hiding secure as per the Definition 6.*

Proof. Since the distribution of normal encryption's ciphertext and special encryption's ciphertext for the index-position-bit tuple $(1, \perp, 0)$ are the same, thus the definition of normal-hiding security follows from construction. □

Lemma 7 (Index-hiding) *If the assumptions 3,4,5 and 6 hold over the bilinear group $\mathbb{BG}$, then our EIPL-IPFE is the index-hiding secure as per the Definition 7.*

Proof. As per the definition of index-hiding game, we show that adversary can not distinguish between the special encryption to the index-position-bit tuple $(i^*, \perp, 0)$ and $(i^*+1, \perp, 0)$. For the index position $i^* = (x^*, y^*)$, the adversary can not query to the key generation oracle for the secret keys.

Now if $y^* = \widetilde{n}$, then we have $i^*+1 = (x^*+1, 1)$ otherwise, $i^*+1 = (x^*, y^*+1)$. Similar to the [BW06, GKW19], we break down the proof into two parts based on whether $y = \widetilde{n}$ or not. To prove this security, we consider following two claims 6 and 7.

Claim 6. For $y^* < \widetilde{n}$, the special encryption corresponding the index-position-value tuple $((x^*, y^*), \perp, 0)$ and $((x^*, y^*+1), \perp, 0)$ are indistinguishable.

Proof of claim 6. To prove the above claim, we consider $2k+1$ sequences of hybrid games as H_0 and $H_{\widetilde{\ell},\widetilde{b}}$ where $\widetilde{\ell} \in [k]$ and $\widetilde{b} \in \{0,1\}$. The hybrid H_0 corresponds to the index-hiding security game where the challenge ciphertext is an encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$ and $H_{\widetilde{\ell},\widetilde{b}}$ is same as H_0 except the column component $C_{y^*,\ell,b}$ for $(\ell, b) \in [\widetilde{\ell}-1] \times \{0,1\}$ and for $\ell = \widetilde{\ell}, b = \widetilde{b}$, we take uniform random element from $\mathbb{G}_p$.

Table 4: Computing column components of the ciphertext in Hybrid $H_{\widetilde{\ell},\widetilde{b}}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell > \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee (y = y^* \wedge \ell < \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b \leq \widetilde{b})$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau} V^{v_{y,\ell,b}\tau}_{\ell,b}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}^{v_{y,\ell,b}}_{\ell,b}$

Here, the hybrid $H_{k,1}$ corresponds to the index-hiding game in which the challenge ciphertext is an encryption of the index-position-bit tuple $(i^*+1, \perp, 0) = ((x^*, y^*+1), \perp, 0)$. It is also required that the hybrid H_0 and $H_{1,0}$ are indistinguishable. In the following, we show that the hybrid $H_{\widetilde{\ell},\widetilde{b}}$ and the hybrid $H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ are indistinguishable. This same proof technique has been used to show all the consecutive hybrids are indistinguishable. By combining all indistinguishability of hybrids, the claim 6 follows.

$H_{\widetilde{\ell},\widetilde{b}} \approx H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$: Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ which can

distinguish between the hybrid $H_{\widetilde{\ell},\widetilde{b}}$ and hybrid $H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \bmod 2}$ with non-negligible advantages. We construct a PPT reduction algorithm $\mathscr{B}$ which breaks the modified-2 D3DH assumption 3 with the same non-negligible advantages as follows:

Let the reduction algorithm $\mathscr{B}$ first receives the modified-2 DBDH assumption 3 challenges from the challenger as

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, D = g_p^{b^2}, T)$$

where T is either g_p^{abc} or a random element in the subgroup $\mathbb{G}_p$ of prime order p. Next, $\mathscr{B}$ receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^m, i^* = (x^*, y^*))$ from the adversary $\mathscr{A}$ where $y^* < \widetilde{n}$. Now, $\mathscr{B}$ generates the master public key by using the modified-2 D3DH instances and sends it to $\mathscr{A}$. Next, the adversary makes adaptively secret keys queries for distinct indices except i^* and sends the challenge message vector $\boldsymbol{v}$ to the challenger. In the following, we show how does $\mathscr{B}$ simulates the master public key and how to answer the queried secret keys and the challenge ciphertext from the challenge instances. Finally, $\mathscr{A}$ outputs its guess, which $\mathscr{B}$ uses to break the modified-2 D3DH assumption 3.

As this reduction plays over the subgroup $\mathbb{G}_p$ with the challenger, thus it can choose any components from the subgroup $\mathbb{G}_q$ by itself. We implicitly set the exponents $r_{p,x^*,j}$, s_{p,x^*} and κ_{p,x^*} as $b \cdot \widetilde{r}_{p,x^*,j}, \widetilde{s}_{p,x^*}/b$ and $\widetilde{\kappa}_{p,x^*}/b^2$ respectively where the exponents $\widetilde{r}_{p,x^*,j}, \widetilde{\kappa}_{p,x^*}\widetilde{s}_{p,x^*}$ are chosen uniformly random from the subgroup $\mathbb{G}_p$ and set $h_p = B = g_p^b, f_p = B^{d_1}$ for some random exponent $d_1 \leftarrow \mathbb{Z}_N$, $t_p = a \cdot b$, $c_{p,y^*,\widetilde{\ell},\widetilde{b}} = c \cdot \widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}}$ where $\widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}} \leftarrow \mathbb{Z}_N$. With these exponents, we correctly simulate the master public key, secret keys and the challenge group elements T that can be programmed in the challenge ciphertext components $C_{y^*,\widetilde{\ell},\widetilde{b}}$.

Public key simulation. Sample two random generators $h_q, f_q \leftarrow \mathbb{G}_q$ such that $h_q = g_q^d, f_q = g_q^{d'}$ and where the exponent $d, d' \leftarrow \mathbb{Z}_N$. Additionally, it chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, \quad & \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad & \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], \quad & \widetilde{r}_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

Samples $\beta \leftarrow \mathbb{Z}_N$ and computes the following public key components for all $x \in [\widehat{n}]$, $j \in [m]$ and $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$ as

$$E_{x,j} = \begin{cases} (g_p g_q)^{\widetilde{r}_{x,j}} \text{ if } x \neq x^*, \\ (B g_q)^{\widetilde{r}_{x,j}} \text{ otherwise.} \end{cases}, \qquad F_{x,j} = \begin{cases} (B h_q)^{\widetilde{r}_{x,j}} \text{ if } x \neq x^*, \\ (D h_q)^{\widetilde{r}_{x,j}} \text{ otherwise.} \end{cases},$$

$$H_{y,\ell,b} = \begin{cases} (C g_q)^{\widetilde{c}_{y,\ell,b}} \text{ if } (y,\ell,b) = (y^*, \widetilde{\ell}, \widetilde{b}), \\ (g_p g_q)^{\widetilde{c}_{y,\ell,b}} \text{ otherwise.} \end{cases}$$

Challenger $\mathscr{B}$ set the master public key as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = g_p^b g_q^d, f = B^{d_1} f_q, E_q = g_q^\beta, Z_q = g_q^{d'\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = g_q^{\beta \widetilde{r}_{x,j}}, F_{q,x,j} = h_q^{\beta \widetilde{r}_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}}, \\ W_{x,j} = e(B^{d_1} f_q, B^{d_1} f_q)^{\psi_{x,j}}, W_{q,x,j} = e(f_q, f_q)^{\beta \psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]}, \\ \{H_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right),$$

All the public key components can be computed using the modified-2 D3DH assumption 3 challenge instances.

Secret Key simulation. In the secret key generation, adversary can not query for the secret key corresponding to the index position $i^* = (x^*, y^*)$ to the key generation oracle. To answer these queries, challenger returns the secret key $\mathsf{sk}_{\boldsymbol{u}}$ corresponding to the tuple $(i = (x, y), \mathsf{id}, \boldsymbol{u})$ as follows:

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x \neq x^*, y \neq y^* \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} (Bg_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x = x^*, (y \neq y^* \vee \mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b}) \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \widetilde{\ell}} c_{y,\ell,\mathsf{id}_\ell}} (Cg_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\widetilde{\ell},\widetilde{b}}} & \text{if } x \neq x^* \wedge (y, \mathsf{id}_{\widetilde{\ell}}) = (y^*, \widetilde{b}) \end{cases}$$

$$K_2 = (B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

Challenge ciphertext simulation. $\mathscr{B}$ makes the simulation of the challenge ciphertext as follows:

$$\begin{aligned} \forall\, j \in [m], \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N,\ \tau \in \mathbb{Z}_N, t_q \leftarrow \mathbb{Z}_N \\ \forall\, x \in [\widehat{n}],\quad e_x, f_x, d_x \leftarrow \mathbb{Z}_N, \widetilde{\kappa}_x, \widetilde{s}_x \leftarrow \mathbb{Z}_N \\ \forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\},\quad \widetilde{w}_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N \end{aligned}$$

Now, the challenger $\mathscr{B}$ simulates the row and column components of the challenge ciphertext as follows:

Table 5: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$.

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{\widetilde{s}_x}$	$F_{q,x,j}^{\widetilde{s}_x \tau}$	$E_q^{\widetilde{s}_x t_q}$	$Z_q^{\widetilde{\kappa}_x t_q}$	$e(g_q, g_q)^{\nu_j} \cdot G_{q,x,j}^{t_q \widetilde{s}_x} \cdot W_{q,x,j}^{t_q \widetilde{\kappa}_x}$
$x = x^*$	$g^{\widetilde{r}_{x,j} \widetilde{s}_x}$	$(Bh_q)^{\widetilde{r}_{x,j} \widetilde{s}_x \tau}$	$(Ag_q^{t_q})^{\widetilde{s}_x}$	$(Af_q^{t_q})^{d_1 \widetilde{\kappa}_x}$	$e(g,g)^{\nu_j} \cdot e(g, Ag_q^{t_q})^{\alpha_{x,j}, \widetilde{s}_x}$ $\cdot e(f, Af_q^{t_q})^{d_1 \psi_{x,j} \widetilde{\kappa}_x}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$(Bh_q)^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	f^{d_x}	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 6: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell > \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\widetilde{w}_{y,\ell,b} \tau}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} \cdot g^{\widetilde{w}_{y,\ell,b}}$
$y = y^* \wedge \ell = \widetilde{\ell} \wedge b = \widetilde{b}$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot T^{\widetilde{c}_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$
$(y < y^*) \vee (y = y^* \wedge \ell < \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot g_p^{\upsilon_{y,\ell,b}}$	$g^{w_{y,\ell,b}}$

After generation of all the ciphertext components, challenger sends these to the adversary $\mathscr{A}$, then $\mathscr{A}$ guess $\mathfrak{b}'$ and sends it to $\mathscr{B}$ and it simply forwards it as its guess to the modified-2 D3DH challenger.

Analysis of Simulation. If $T = g_p^{abc}$, then $\mathscr{B}$ simulates the view of the hybrid $H_{\widetilde{\ell},\widetilde{b}}$ otherwise if T is randomly sampled from the subgroup $\mathbb{G}_p$, then $\mathscr{B}$ simulates the view of the hybrid $H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$. So if $\mathscr{A}$ wins with the advantage $\epsilon(\cdot)$, then $\mathscr{B}$ breaks the modified-2 D3DH assumption with same advantage.

Claim 7. If $y^* = \widetilde{n}$, the special encryption to the index-position-bit tuple $((x^*, y^*), \perp, 0)$ and $((x^*+1, 1), \perp, 0)$ are indistinguishable.

Proof of claim 7. To prove the above claim, we consider a sequence of hybrids. In the following, we discuss about these hybrids.

Hybrid 1. This hybrid corresponds to the index-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$ for $y^* = \widetilde{n}$.

Hybrid 2. The hybrid is exactly same as hybrid 1 except that the challenge ciphertext is a special encryption to index-position-bit tuple $(i^* = (x^*, y^* + 1), \perp, 0)$ for $y^* = \widetilde{n}$. In general, as $y^* = \widetilde{n}$, the special-encryption algorithm does not encrypt to the position $(x^*, y^* + 1)$generally. However, this the algorithm can be naturally extended to encrypt such specific position.

Hybrid 3. Hybrid 3 is identical to the previous hybrid 2 except that the row component $I_{x,j}$ for $x = x^*$ of the special encryption as mentioned in the following Table 7.

Table 7: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t}$	$Z_q^{\kappa_x t}$	$e(g_q, g_q)^{v_j} \cdot G_{q,x,j}^{t s_x} \cdot W_{q,x,j}^{t \kappa_x}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$g^{s_x t}$	$f^{\kappa_x t}$	$e(g,g)^{v_j} \cdot G_{x,j}^{t s_x} \cdot W_{x,j}^{t \kappa_x} L$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau v_j}$	g^{e_x}	f^{d_x}	$e(g,g)^{f_x \phi_j} e(f,f)^{f_x \phi_j}$

where $L = e(g_p, g)^z$ for $z \in \mathbb{Z}_p$ is randomly chosen.

Hybrid 4. Hybrid 4 is identical to hybrid 3 except that the row component of challenge ciphertext as in Table 7, where $L = e(g,g)^z$ with z as a random exponent from $\mathbb{Z}_N$.

Hybrid 5. Hybrid 5 is same as hybrid 4 except that the row components in the Table 8 which we mention below.

Table 8: Computing row components for $x \in [\widehat{n}]$, $j \in [m]$ in Hybrid 5

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t}$	$Z_q^{\kappa_x t}$	$e(g_q, g_q)^{v_j} \cdot G_{q,x,j}^{t s_x} \cdot W_{q,x,j}^{t \kappa_x}$
$x \le x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau v_j}$	g^{e_x}	f^{d_x}	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Hybrid 6. The hybrid 6 is similar to the hybrid 5 except that the column components to the index-position-bit tuple $((x^*, y^* = 1), \ell^* = \perp, b^* = 0)$ of the challenge ciphertext.

Hybrid 7. The hybrid 7 corresponds to the index-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $((x^* + 1, 1), \perp, 0)$ for $y^* = \widetilde{n}$.

In the following, we have proved that adversary's advantage for all the consecutive hybrids is negligible in the security parameter which completes the proof of the claim 7.

Hybrid 1 $\approx$ Hybrid 2: The indistinguishability proof of the hybrid 1 and hybrid 2 is identical to the claim 6.

Hybrid 2 ≈ Hybrid 3: Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ that distinguishes between the above two hybrids with $\epsilon(\cdot)$ advantages. Then we construct a PPT reduction algorithm which breaks the DBDH assumption with the same advantages as follows:

The reduction algorithm $\mathscr{B}$ first receives the DBDH challenge from the challenger as

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, T = e(g_p, g)^z)$$

where z is either abc or a random element from $\mathbb{Z}_N$. Next, in the setup phase, adversary receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^m, , i^* = (x^*, y^*))$ from adversary $\mathscr{A}$ satisfying the condition $y^* = \widetilde{n}$. We now apply the reduction games with its challenger in the subgroup $\mathbb{G}_p$, thus it can choose all the elements from the subgroup $\mathbb{G}_q$ by itself. Now, $\mathscr{B}$ generates the master public key using the instances of DBDH assumption and sends it to the adversary $\mathscr{A}$. Next, the adversary makes the polynomially many secret key queries to the key generation oracle corresponding to any index except i^* and vector $\boldsymbol{u}$. In the following, we show how $\mathscr{B}$ simulate the master public key, secret keys and challenge ciphertext from the DBDH instances. Finally, $\mathscr{A}$ outputs its guess, which is used to break the DBDH assumption. For $x = x^*$, our approach is to implicitly set the exponents $r_{p,x^*,j} = b\mathfrak{r}_j, \alpha_{p,x^*,j} = abk\mathfrak{r}_j, t_p = c$ for all $\mathfrak{r}_j \leftarrow \mathbb{Z}_N$. Additionally, it implicitly sets $c_{y,\ell,b} = \widetilde{c}_{p,y,\ell,b} - a$ for all $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$. According to the exponents as given above, the challenger simulates the master public key components, the secret keys and the challenge ciphertext components. In the challenge ciphertext, the challenge group elements T can be programmed in the challenge ciphertext component $I_{x^*,j}$.

Public key simulation. It chooses random exponents d, d_1 from $\mathbb{Z}_N$ such that $h = g^d, f = g^{d_1}$. To generate the master public key, it also chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, &\quad \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], &\quad \widetilde{r}_{x,j}, \widetilde{\alpha}_{x,j}, \widetilde{\psi}_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

Also, it samples $d, \beta \leftarrow \mathbb{Z}_N$ and for all $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$ computes

$$E_{x,j} = \begin{cases} (g_p g_q)^{\widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ B^{\mathfrak{r}_j} g_q^{\widetilde{r}_{x,j}} & \text{otherwise.} \end{cases}, \quad F_{x,j} = \begin{cases} (g_p g_q)^{d\widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ B^{d\mathfrak{r}_j} g_q^{d\widetilde{r}_{x,j}} & \text{otherwise.} \end{cases},$$

$$G_{x,j} = \begin{cases} e(g_p g_q, g_p g_q)^{\widetilde{\alpha}_{x,j}} & \text{if } x \neq x^*, \\ e(A,B)^{k\mathfrak{r}_j} e(g_q, g_q)^{\widetilde{\alpha}_{x,j}} & \text{otherwise.} \end{cases}$$

Sets the master public key

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = g^d, f = g^{d_1}, Z_q = f_q^\beta, E_q = g_q^\beta, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j}, \\ E_{q,x,j} = g_q^{\beta\widetilde{r}_{x,j}}, F_{q,x,j} = h^{\beta\widetilde{r}_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta\widetilde{\alpha}_{x,j}}, \\ W_{x,j} = e(g,g)^{d_1^2\widetilde{\psi}_{x,j}}, W_{q,x,j} = e(f_q, f_q)^{\beta d_1^2 \widetilde{\psi}_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b} = A^{-1} g^{\widetilde{c}_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

Secret key simulation. The adversary $\mathscr{A}$ can not query to the challenge index position $i^* = (x^*, y^*)$. Challenger $\mathscr{B}$ returns secret key to the adversary associated to the query tuple $(i = (x, y), \mathsf{id}, \boldsymbol{u})$ as

$$K_1 = \begin{cases} g^{\langle \widetilde{\boldsymbol{\alpha}}_x, \boldsymbol{u}\rangle + \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} A^{-k\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle} & \text{if } x \neq x^*, \\ g_q^{\langle \widetilde{\boldsymbol{\alpha}}_x, \boldsymbol{u}\rangle + \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} B^{\langle \mathfrak{r}, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{otherwise.} \end{cases}$$

$$K_2 = g^{d_1 \langle \widetilde{\boldsymbol{\psi}}_x, \boldsymbol{u}\rangle} \text{ for all } x$$

Challenge ciphertext simulation. The challenger $\mathscr{B}$ can compute all the column components corresponding to the index position $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0, 1\}$ on its own since $\mathbb{G}_p$ subgroup components are random in $C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}$ terms, and for computing over the subgroup $\mathbb{G}_q$, all the required components challenger $\mathscr{B}$ already knows. For $x < x^*$, all the row components $R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}$ are chosen random but for $x > x^*$ all the row components are formed over the subgroup $\mathbb{G}_q$ which $\mathscr{B}$ knows. The challenger $\mathscr{B}$ will simulate the challenge ciphertext for $x = x^*$ as follows:

Table 9: Computing row component of the ciphertext for $x = x^*$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x = x^*$	$(B^{\mathfrak{r}_j} g_q^{\widetilde{r}_{x,j}})^{\widetilde{s}_x}$	$(B^{\mathfrak{r}_j} g_q^{\widetilde{r}_{x,j}})^{d\tau\widetilde{s}_x}$	$(Cg_q^{t_q})^{\widetilde{s}_x}$	$(Cg_q^{t_q})^{d_1\widetilde{\kappa}_x}$	$e(g,g)^{v_j} \cdot e(g_q, g_q)^{\widetilde{\alpha}_{x,j}\widetilde{s}_x t_q}$ $\cdot e(C, g_p)^{d_1^2 \widetilde{\psi}_{x,j}\widetilde{\kappa}_x} \cdot T^{k\mathfrak{r}_j\widetilde{s}_x}$ $\cdot e(g_q, g_q)^{d_1^2\widetilde{\psi}_{x,j}\widetilde{\kappa}_x t_q}$

where the exponents $\widetilde{s}_{x^*}, \widetilde{\kappa}_{x^*}, \tau \in \mathbb{Z}_N$ and $t_q \leftarrow \mathbb{Z}_N$ are sampled uniformly at random. Finally, $\mathscr{B}$ gets the guess bit $\mathfrak{b}'$ from $\mathscr{A}$ and it simply forwards it to the DBDH challenger.

Analysis of simulation. If $T = e(g_p, g)^{abc}$, then $\mathscr{B}$ simulates the view of hybrid 2 else the adversary's view same as hybrid 3 for $T = e(g_p, g)^z$ for any random z from $\mathbb{Z}_N$. Therefore, if $\mathscr{B}$ breaks the DBDH assumption 3 with $\epsilon(\cdot)$ advantages then $\mathscr{A}$ wins the game with same advantages.

Hybrid 3 ≈ Hybrid 4: To show the indistinguishability of two hybrids 3 and 4, we use similar proof technique of [BW06, GKW19]. Here, we discuss the underlying approaches. Let us consider that $\mathscr{B}$ receives the challenges of Bilinear Subgroup Decisional (BSD) assumption 5 from the challenger consisting a bilinear group $\mathbb{BG}, e(T, g)$ where T is either a random element from the subgroup $\mathbb{G}_p$ or an uniform element from the group $\mathbb{G}$. Then, $\mathscr{B}$ computes all the components of the master public key mpk honestly and forwarded it to the adversary $\mathscr{A}$. After seeing mpk, adversary can query for a secret key to the key generation oracle. Finally, $\mathscr{B}$ computes all the challenge ciphertext components honestly except the value $I_{x^*,j} = e(g,g)^{v_j^{(\mathfrak{b})}} \cdot G_{x^*,j}^{s_{x^*}t} \cdot e(g, T) \cdot e(f, f)^{\psi_{x,j}\kappa_x t}$ where T is taken from BSD challenge assumption 5. If $T \leftarrow \mathbb{G}_p$, then simulator's view is same as hybrid 3 otherwise if $T \leftarrow \mathbb{G}$ then $\mathscr{B}$ perfectly simulates as hybrid 4. Therefore, if $\mathscr{A}$ wins with the advantage $\epsilon(\cdot)$, then $\mathscr{B}$ breaks the assumption 3 with same advantages $\epsilon(\cdot)$.

Hybrid 4 ≈ Hybrid 5: Suppose on contrary, there exists PPT adversary $\mathscr{A}$ that distinguishes between the hybrid 4 and hybrid 5 with the non-negligible advantage $\epsilon(\cdot)$, then we construct a PPT reduction algorithm which breaks the assumption 6 with the same non-negligible advantages as follows.

The reduction algorithm $\mathscr{B}$ first receives the R3DH challenge instances from the challenger as

$$(\mathbb{BG}, \mathfrak{g}_p \in \mathbb{G}_p, \mathfrak{g}_q \in \mathbb{G}_q, A = \mathfrak{g}_q^a, B = \mathfrak{g}_p^{\widetilde{a}} \cdot \mathfrak{g}_q^{a^2}, C = \mathfrak{g}_p^{\widetilde{c}} \cdot \mathfrak{g}_q^c, D = \mathfrak{g}_p^{\widetilde{a}\widetilde{c}}, T)$$

where T is either $\mathfrak{g}_q^{a^2 c}$ or a random element from the subgroup $\mathbb{G}_q$. Next, the challenger $\mathscr{B}$ receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^m, i^* = (x^*, y^*))$ from the adversary $\mathscr{A}$. Then, $\mathscr{B}$ generates the master public keys and sends it to adversary. After seeing the public parameters, $\mathscr{A}$ can adaptively query to the key generation oracle corresponding to the tuple $(i = (x, y), \boldsymbol{u}, \mathsf{id})$. Then the adversary uniformly chooses a challenge message vector pair $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ and sends it to $\mathscr{B}$. Challenger randomly chooses a bit $\mathfrak{b} \in \{0,1\}$ and generates the challenge ciphertext $\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}$. In the following, we describe how does the challenger simulate the master public key, secret keys and the challenge ciphertext using the R3DH assumption 6 instances. Finally, the adversary $\mathscr{A}$ outputs a guess bit which breaks the R3DH assumption 6. Since the reduction plays the game with challenger in the subgroup $\mathbb{G}_q$, so it choose most components in the subgroup $\mathbb{G}_p$ by itself. Although, the challenge components of B, C are some part belonging to the subgroup $\mathbb{G}_p$. Thus some of their exponents will also implicitly depends on the $\widetilde{a}$ and $\widetilde{c}$ terms. In the following, we implicitly set the exponents as

$$\begin{aligned} &g_p = \mathfrak{g}_p \quad g_q = A, \qquad r_{q,x^*,j} = \frac{\widetilde{r}_{q,x^*,j}}{a}, \qquad r_{p,x^*,j} = \widetilde{r}_{p,x^*,j}, \\ &s_{q,x^*} = c,\ s_{p,x^*} = \widetilde{c},\ \kappa_{q,x^*} = 1/a,\ \kappa_{p,x^*} = 1/\widetilde{a}, \qquad t_q = a, \qquad t_p = \widetilde{a}, \end{aligned}$$

$$s_x = \widetilde{s}_x / a; \kappa_x = \widetilde{\kappa}_x / a \text{ for all } x \in [\widehat{n}] - \{x^*\}$$

where $\widetilde{r}_{p,x^*,j}, \widetilde{r}_{q,x^*,j} \leftarrow \mathbb{Z}_N$ and for all $x \in [\widehat{n}] - \{x^*\}, \widetilde{s}_x, \widetilde{\kappa}_x \leftarrow \mathbb{Z}_N$. Additionally, the reduction algorithm samples the exponents uniformly random from $\mathbb{Z}_N$. As the reduction algorithm does not know the factorization, so at any instant we do not sample the exponents from $\mathbb{G}_p$ and $\mathbb{G}_q$ individually. So instead of this we sample an exponent directly from $\mathbb{Z}_N$ and it need to make sure that the distributions are not disturbed.

Public key simulation. To generate the public key it chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, &\quad c_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], &\quad \widetilde{r}_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

Also, we choose the exponents $d, d_1, \beta \leftarrow \mathbb{Z}_N$ to compute all the remaining components of the master public key for all $x \in [\widehat{n}], j \in [m]$ and $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$ and set $h = g^d, f - g6d_1$.

$$E_{q,x,j} = \begin{cases} A^{\beta \widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ \mathfrak{g}_q^{\beta \widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases}, \quad E_{x,j} = \begin{cases} (\mathfrak{g}_p A)^{\widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ (\mathfrak{g}_p \mathfrak{g}_q)^{\widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases}, \quad F_{q,x,j} = E_{q,x,j}^d, \quad F_{x,j} = E_{x,j}^d,$$

$$\begin{aligned} G_{q,x,j} &= e(A,A)^{\beta \alpha_{x,j}}, \qquad W_{q,x,j} = e(A,A)^{\beta d_1^2 \psi_{x,j}}, \\ G_{x,j} &= e(\mathfrak{g}_p A, \mathfrak{g}_p A)^{\alpha_{x,j}}, \quad W_{x,j} = e(\mathfrak{g}_p A, \mathfrak{g}_p A)^{d_1^2 \psi_{x,j}}. \end{aligned}$$

So, its master public key set as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = \mathfrak{g}_p A, h = g^d, f = g^{d_1}, E_q = A^\beta, Z_q = A^{d_1\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j}, W_{x,j}, \\ E_{q,x,j}, F_{q,x,j}, G_{q,x,j}, W_{q,x,j} \end{array} \right\}_{x\in[\widehat{n}], j\in[m]}, \\ \{H_{y,\ell,b} = (\mathfrak{g}_p A)^{c_{y,\ell,b}}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b)\in[k]\times\{0,1\}} \end{array} \right)$$

Secret key simulation. In the secret key generation, the adversary $\mathscr{A}$ is not allowed to query for the challenge index position $i^* = (x^*, y^*)$ to the key generation oracle. $\mathscr{B}$ answers the secret key associated to the index position $i = (x, y)$, identity id and the predicate vector $\boldsymbol{u}$ as given below.

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (\mathfrak{g}_p A)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} & \text{if } x \neq x^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (\mathfrak{g}_p \mathfrak{g}_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} & \text{otherwise.} \end{cases}$$

$$K_2 = (\mathfrak{g}_p A)^{d_1 \langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \text{ for all } x$$

Challenge ciphertext simulation. To generate the challenge ciphertext, $\mathscr{B}$ chooses the random exponents as follows:

$$\begin{aligned} \forall\, j \in [m], \sigma_j, \nu_j, \phi_j &\leftarrow \mathbb{Z}_N, \quad \tau \in \mathbb{Z}_N, \\ \forall\, x \in [\widehat{n}], \quad e_x, f_x, d_x &\leftarrow \mathbb{Z}_N, \widetilde{s}_x, \widetilde{\kappa}_x \leftarrow \mathbb{Z}_N, \\ \forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad \widetilde{w}_{y,\ell,b}, \widetilde{v}_{y,\ell,b} &\leftarrow \mathbb{Z}_N. \end{aligned}$$

Now, $\mathscr{B}$ computes following row and column components as follows:

Table 10: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$\mathfrak{g}_q^{\beta \widetilde{r}_{x,j} \widetilde{s}_x}$	$\mathfrak{g}_q^{\beta\tau d \widetilde{r}_{x,j} \widetilde{s}_x}$	$A^{\beta \widetilde{s}_x}$	$A^{d_1 \beta \widetilde{\kappa}_x}$	$e(g_q, g_q)^{\nu_j} e(A,A)^{\beta \alpha_{x,j} \widetilde{s}_x} e(A,A)^{\beta d_1^2 \widetilde{\kappa}_x \psi_{x,j}}$
$x = x^*$	$C^{\widetilde{r}_{x,j}}$	$C^{\widetilde{r}_{x,j}\tau d}$	DT	$\mathfrak{g}_p^{d_1} A^{d_1}$	$e(g,g)^{\phi_j f_x} e(g,g)^{d_1^2 \phi_j f_x}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$g^{d \widetilde{s}_x \tau \nu_j}$	g^{e_x}	$g^{d_1 d_x}$	$e(g,g)^{f_x \phi_j} e(g,g)^{d_1^2 \phi_j f_x}$

Table 11: Column components computation for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$\forall\, y \in [\widetilde{n}]$	$B^{c_{y,\ell,b}} h^{\widetilde{w}_{y,\ell,b}\tau} \mathfrak{g}_p^{\widetilde{v}_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$

Analysis of simulation. For $T = \mathfrak{g}_q^{a^2 c}$, $\mathscr{B}$ simulates the hybrid 4, otherwise if T is randomly chosen from the group $\mathbb{G}_q$ then $\mathscr{B}$ simulates the view of hybrid 5. Therefore, for the case of $x = x^*$ and $x < x^*$, the adversarial view of the hybrids 4 and 5 will be indistinguishable. If $\mathscr{B}$ breaks R3DH assumption 6 with the advantage $\epsilon(\cdot)$ then $\mathscr{A}$ wins with same advantage.

Hybrid 5 ≈ Hybrid 6: To prove the hybrid 5 and 6 are indistinguishable, let us consider $(2\widetilde{n}k+1)$ sub-hybrid $H_0, H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ for $(\widetilde{y}, \widetilde{\ell}, \widetilde{b}) \in [\widetilde{n}] \times [k] \times \{0,1\}$. In this game the sub-hybrid H_0 corresponds to the hybrid 5 as described above. Now the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ is same as the hybrid H_0 except the column components in the challenge ciphertext $C_{y,\ell,b}$ for $y < \widetilde{y}$ and $(y,\ell,b) \in \{\widetilde{y}\} \times [\widetilde{\ell} - 1] \times \{0,1\}$ and for $y = \widetilde{y}, \ell = \widetilde{\ell}, b < \widetilde{b}$ have a random element in the subgroup $\mathbb{G}_p$. The column components are generated as described below in the Table 12.

Table 12: Computing column components of ciphertext for the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell > \widetilde{\ell}) \vee$ $(y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b \geq \widetilde{b})$	$H_{y,\ell,b}^{t} \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell < \widetilde{\ell}) \vee$ $(y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$H_{y,\ell,b}^{t} \cdot h^{w_{y,\ell,b}\tau} \cdot V_{\ell,b}^{v_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$

In the following, we show that the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ is indistinguishability with the sub-hybrid $H_{\widetilde{y},\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$. Note that, the sub hybrid $H_{1,1,0}$ is identical with the main hybrid 6. It is also required that the sub-hybrid $H_0 \approx H_{\widetilde{n},\ell,1}$ and sub-hybrid $H_{\widetilde{y},k,1} \approx H_{\widetilde{y}+1,1,0}$. We show that the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ and $H_{\widetilde{y},\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ are indistinguishable whose techniques are similar to prove the indistinguishability of all the consecutive sub-hybrids for all $(\widetilde{y},\widetilde{\ell},\widetilde{b})$. By combining all the proof of sub-hybrids, the main two hybrids (hybrid 5 and hybrid 6) are indistinguishable.

Sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}} \approx$ *Sub-hybrid* $H_{\widetilde{y},\widetilde{y}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$: Suppose on contrary, there exist a PPT adversary that distinguish between the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ and sub-hybrid $H_{\widetilde{y},\widetilde{y}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ with the non-negligible advantage $\epsilon(\cdot)$. We construct a PPT reduction algorithm $\mathscr{B}$ which can break the D3DH assumption 3 with same non-negligible advantage as described below. From the challenger, the reduction algorithm $\mathscr{B}$ receives the following instances as

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, T)$$

where T is either g_p^{abc} or a random element in the subgroup $\mathbb{G}_p$ of prime order p. Next, it receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^m, i^* = (x^*, y^*))$ from the adversary $\mathscr{A}$ where $y^* = \widetilde{n}$. Now, $\mathscr{B}$ generates the master public key using the D3DH assumption 3 instances and sends it to the adversary. Then, the adversary can adaptively make queries for the secret keys of distinct indices except i^* and sets the challenge ciphertext $\mathsf{ct}_{\boldsymbol{\nu}}^{(\mathsf{b})}$. In the following, $\mathscr{B}$ simulates the master public key, secret keys and the challenge ciphertext. Finally, $\mathscr{A}$ outputs its guess, which $\mathscr{B}$ uses to break the assumption 3. Since, the reduction plays with the challenger over the subgroup $\mathbb{G}_p$, thus it can choose everything from the subgroup $\mathbb{G}_q$ by itself. It now implicitly sets the exponents as $t_p = a \cdot b$, $c_{p,\widetilde{y},\widetilde{\ell},\widetilde{b}} = c \cdot \widetilde{c}_{p,\widetilde{y},\widetilde{\ell},\widetilde{b}}$ where the exponent $\widetilde{c}_{p,\widetilde{y},\widetilde{\ell},\widetilde{b}}$ is chosen uniformly at random and also set and $h_p = B = g_p^b$. By using these exponents, $\mathscr{B}$ simulates the master public key, secret keys and the group elements T can be programmed in the challenge ciphertext components $C_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$.

Public key simulation. To simulate the master public key, $\mathscr{B}$ sets $h_q = g_q^d$ where $d \leftarrow \mathbb{Z}_N$ and chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, &\quad \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], &\quad r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

It also samples $\beta, d_1, d_2 \leftarrow \mathbb{Z}_N$ and sets $g = g_p g_q$, $h = h_p h_q = B h_q = g_p^b \cdot g_q^d$, $f = f_p f_q = g_p^{d_1} g_q^{d_2}$. All the components of the master public keys are generated as follows:

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = B h_q, f = g^d, E_q = g_q^\beta, Z_q = f_q^\beta = g_q^{\beta d_1} \\ \left\{ \begin{array}{c} E_{x,j} = g^{r_{x,j}}, F_{x,j} = h^{r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = g_q^{\beta r_{x,j}}, F_{q,x,j} = h_q^{\beta r_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}}, \\ W_{x,j} = e(f,f)^{\psi_{x,j}}, W_{q,x,j} = e(g_q, g_q)^{\beta d_1^2 \psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

All the components can be generated using the challenge D3DH instances of assumption 3.

Secret key simulation. $\mathscr{B}$ answers the secret keys associated to the index $i = (x, y)$, identity id and the predicate vector $\boldsymbol{u}$ as given below.

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \widetilde{\ell}} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} (C g_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{\widetilde{y},\widetilde{\ell},\widetilde{b}}} & \text{if } (y, \mathsf{id}_{\widetilde{\ell}}) = (\widetilde{y}, \widetilde{b}), \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{otherwise.} \end{cases}$$

$$K_2 = g^{d \langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

In the secret key query phase, the adversary $\mathscr{A}$ is not allowed to key query corresponding to the challenge index position $i^* = (x^*, y^*)$.

Challenge ciphertext simulation. Challenger $\mathscr{B}$ chooses the random exponents as follows

$$\begin{aligned} \forall\, j \in [m], \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N,\ \tau, t_q \in \mathbb{Z}_N, \\ \forall\, x \in [\widehat{n}], \quad \kappa_x, e_x, f_x, d_x, s_x \leftarrow \mathbb{Z}_N \\ \forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad \widetilde{w}_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N \end{aligned}$$

Now, for all $x \in [\widehat{n}]$, $j \in [m]$, challenger can compute following row and column components as in Table 13, 14 respectively.

Table 13: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t_q}$	$Z_q^{\kappa_x t_q}$	$e(g_q, g_q)^{\upsilon_j} \cdot G_{q,x,j}^{s_x t_q} \cdot W_{q,x,j}^{t_q \kappa_x}$
$x \leq x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	f^{d_x}	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 14: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell > \widetilde{\ell}) \vee (y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\widetilde{w}_{y,\ell,b} \tau}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} g^{\widetilde{w}_{y,\ell,b}}$
$y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b = \widetilde{b}$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\tau \widetilde{w}_{y,\ell,b}} T^{\widetilde{c}_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$
$(y < \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell < \widetilde{\ell}) \vee (y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\tau \widetilde{w}_{y,\ell,b}} g_p^{\upsilon_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$

Analysis of simulation. For $T = g_p^{abc}$, then $\mathscr{A}$ gets the view of the challenge ciphertext as the sub-hybrid $H_{\widetilde{y},\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$, otherwise for any other random group elements from the sub-group $\mathbb{G}_p$, the adversary $\mathscr{A}$ gets the view of sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$. If the adversary $\mathscr{A}$ wins with an advantage $\epsilon(\cdot)$, then $\mathscr{B}$ can break the modified-2 D3DH assumption 3 with the same advantages.

Hybrid 6 ≈ Hybrid 7: Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ that can distinguish the hybrid 6 and hybrid 7 with non-negligible advantage $\epsilon(\cdot)$. Now, we construct a PPT reduction algorithm $\mathscr{B}$ that breaks the DHSD assumption 4 with same advantage as follows:

The reduction algorithm $\mathscr{B}$ first receives the challenge instances of assumption 4 from the challenger as

$$(\mathbb{BG}, g = g_p g_q, h = h_p h_q, A = g_q^a, B = h_q^a, C = g^b g_p^c, D = h^b, T)$$

where T is either sampled as $T = g_q^d$ or $T = g^d$ with $d \leftarrow \mathbb{Z}_N$. Next, $\mathscr{B}$ receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^m, i^* = (x^*, y^*))$ from the adversary for $y^* = \widetilde{n}$. In the following, $\mathscr{B}$ simulates the master public key from the DHSD instances of assumption 4 and sends it to the adversary $\mathscr{A}$. Then, $\mathscr{A}$ creates adaptively secret key queries for the distinct indices except i^* and sets the challenge ciphertext $\mathsf{ct}_{\boldsymbol{v}}^{(\mathsf{b})}$. Using the instances of assumption 4, $\mathscr{B}$ successfully simulates the secret keys and the challenge ciphertext. Finally, $\mathscr{A}$ outputs its guess, which $\mathscr{B}$ uses to break the DHSD assumption 4. In this proof, the reduction plays with its challenger in the subgroup $\mathbb{G}_p$ thus it can choose everything from the subgroup $\mathbb{G}_q$ by itself. To prove this indistinguishability, $\mathscr{B}$ first implicitly sets the random exponents as

$$\beta = a,\ s_{x^*+1} = d \cdot \widetilde{s}_{x^*+1},\ \gamma_{\ell,b} = c \cdot \widetilde{\gamma}_{\ell,b},\ \delta_{\ell,b} = b \cdot \widetilde{\gamma}_{\ell,b} + \widetilde{\delta}_{\ell,b}$$

where the exponents $\widetilde{\gamma}_{\ell,b}, \widetilde{\delta}_{\ell,b}$ are uniformly chosen from $\mathbb{Z}_N$. Also, $\mathscr{B}$ implicitly sets $h^\tau = g^\pi$ where π be any random exponent from $\mathbb{Z}_N$. Using the DHSD instances, the challenge group element T can be programmed to compute the row component $I_{x^*+1.j}$.

Public key simulation. Challenger $\mathscr{B}$ chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \widetilde{\delta}_{\ell,b} \leftarrow \mathbb{Z}_N, \widetilde{\gamma}_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, &\quad c_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], &\quad r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

It chooses randomly an exponent $d' \leftarrow \mathbb{Z}_N$ and sets $f_q = g_q^{d'}$. In the following, $\mathscr{B}$ computes the master public key using the DHSD challenge instances as follows:

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g, h, f, E_q = A, Z_q = A^{d'}, \\ \left\{ \begin{array}{c} E_{x,j} = g^{r_{x,j}}, F_{x,j} = h^{r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = A^{r_{x,j}}, F_{q,x,j} = B^{r_{x,j}}, G_{q,x,j} = e(A,g_q)^{\alpha_{x,j}}, \\ W_{x,j} = e(f,f)^{\psi_{x,j}}, W_{q,x,j} = e(A,g_q)^{d' 2 \psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b} = g^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\widetilde{\delta}_{\ell,b}} C^{\widetilde{\gamma}_{\ell,b}}, V_{\ell,b} = h^{\widetilde{\delta}_{\ell,b}} D^{\widetilde{\gamma}_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

Secret Key Simulation. In this phase, $\mathscr{B}$ answers the queried secret keys $\mathsf{sk}_{\boldsymbol{u}}$ for the tuple $(i = (x,y), \mathsf{id}, \boldsymbol{u})$ where $i \neq i^*$ as given below

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle}, \quad K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

Challenge ciphertext simulation. $\mathscr{B}$ chooses the random exponents as follows:

$$\begin{aligned} \forall j \in [m], \ \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N, \ \pi \in \mathbb{Z}_N, t \leftarrow \mathbb{Z}_N, \\ \forall x \in [\widehat{n}], \quad e_x, f_x, \widetilde{s}_x, d_x, \kappa_x \leftarrow \mathbb{Z}_N, \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad w_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N, \end{aligned}$$

In the following Table 15, 16, we compute the row and column component of the challenge ciphertext.

Table 15: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^* + 1$	$E_{q,x,j}^{\widetilde{s}_x}$	$F_{q,x,j}^{\widetilde{s}_x \pi}$	$E_q^{\widetilde{s}_x t}$	$Z_q^{t\kappa_x}$	$e(g_q, g_q)^{\upsilon_j} \cdot G_{q,x,j}^{\widetilde{s}_x t} \cdot W_{q,x,j}^{t\kappa_x}$
$x = x^* + 1$	$T^{\widetilde{s}_x r_{x,j}}$	$T^{\widetilde{s}_x r_{x,j} \pi}$	$T^{\widetilde{s}_x t}$	$f^{t\kappa_x}$	$e(g,g)^{\upsilon_j} \cdot e(T,g)^{\widetilde{s}_x \alpha_{x,j} t} \cdot W_{x,j}^{t\kappa_x}$
$x < x^* + 1$	$g^{\widetilde{s}_x \sigma_j}$	$h^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	f^{d_x}	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 16: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$\forall (y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}$	$H_{y,\ell,b}^t \cdot g^{w_{y,\ell,b}\pi}$	$g^{w_{y,\ell,b}}$

After seeing the challenge ciphertext, $\mathscr{B}$ receives a guess bit $\mathfrak{b}'$ from $\mathscr{A}$ which is forwarded as its guesses to the DHSD challenger.

Analysis of simulation. If $T = g_q^d$, then $\mathscr{B}$ simulates the view of hybrid 6 otherwise, if T is randomly chosen element from the group $\mathbb{G}$. Then $\mathscr{B}$ simulates the view same as hybrid 7. Therefore, if $\mathscr{A}$ wins the with non-negligible advantages $\epsilon(\cdot)$ then $\mathscr{B}$ breaks the DHSD assumption 4 with same advantage. Hence, proof of the Lemma 7 is complete. □

This concludes the proof of index-hiding security. □

Lemma 8 (Lower identity-hiding) *If the assumptions 3 holds over the bilinear group $\mathbb{BG}$, then our EIPL-IPFE satisfies the lower identity-hiding security as per the Definition 8.*

Proof. The lower-identity security requires that no PPT adversary can distinguish between the special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \ell^*, b^*)$ and $(i^* = (x^*, y^*), \perp, 0)$ with non-negligible advantages and given the adversary does not have a secret key for $(i^*, \mathsf{id}, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = b^*$. This proof technique is similar to that used in Lemma 6. Here, we just exclude the intermediate hybrids as mention in the previous proof. Let $(i^* = (x^*, y^*), \ell^*, b^*)$ be the challenge tuple provided by the adversary $\mathscr{A}$. The hybrid H_{ℓ^*, b^*} corresponds to the exactly the same as the lower identity-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \ell^*, b^*)$ and similarly hybrid $H_{0,1}$ is the same as the lower identity-hiding game for the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$. So, the indistinguishability proof of the special encryption for the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$ and $(i^* = (x^*, y^*), \ell^*, b^*)$ are similar to the Lemma 6.

In the following, we discuss about the secret key simulation where the reduction algorithm $\mathscr{B}$ answers the all permissible secret keys corresponding to the tuple $(i, \mathsf{id}, \boldsymbol{u})$ as per the lower identity security game. From the security restriction of this game, the adversary can not query for the secret key corresponds to the index i^* and the identity id such that

$\mathsf{id}_{\ell^*} = b^*$. So all the key queries are in the form either $i \neq i^*$ or $\mathsf{id}_{\ell^*} \neq b^*$.

Secret key simulation. The secret key $\mathsf{sk}_{\boldsymbol{u}}$ corresponding to the tuple $(i, \mathsf{id}, \boldsymbol{u})$ are as follows:

– For $i \neq i^*$ i.e., $x \neq x^* \wedge y \neq y^*$

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}, \; K_2 = (B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

– For, $(x = x^*) \wedge (y \neq y^* \vee \mathsf{id}_{\ell^*} \neq b^*)$

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (B g_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}, \; K_2 = (B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

– For, $(x \neq x^*) \wedge (y = y^* \wedge \mathsf{id}_{\ell^*} = b^*)$

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle + \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (C g_q)^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,\mathsf{id}_{\ell^*}}} \; K_2 = (B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}$$

Note that, in the second case, the adversary can ask for the secret keys corresponding to the index i^*, which can be answered as long as the queried identity must satisfy $\mathsf{id}_{\ell^*} \neq 0$. Since the lower identity-hiding game requires the adversary not to ask the key queries in this form (i^*, id) with $\mathsf{id}_{\ell^* \neq b^*}$, then the reduction algorithm can perfectly simulate the low identity-hiding game so that this scheme satisfies the security of the lower identity-hiding security assuming the modified-2 D3DH assumption 3 holds.

It completes the proof of lower identity-hiding. □

Lemma 9 (Upper identity-hiding) *If the assumptions 3,4,5 and 6 hold over the bilinear group $\mathbb{BG}$, then our EIPL-IPFE satisfies the upper-identity hiding security as per the Definition 9.*

Proof. We will prove this Lemma 9, via a sequence of hybrid games as discuss below. Recall that, the upper-identity security requires that no PPT adversary can distinguish between the special encryption to the index-position-bit tuple (i^*, ℓ^*, b^*) and $(i^* + 1, \perp, 0)$ with non-negligible advantages and given the adversary does not have a secret key for the tuple $(i^*, \mathsf{id}, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = 1 - b^*$.

$\mathbf{Hyb_1}$: The hybrid corresponding to the upper identity-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \ell^*, b^*)$.

$\mathbf{Hyb_2}$: The hybrid is the similar as the Hyb_1 except that the columns components as the table below.

Table 17: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0, 1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell \neq \ell^*)$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee ((y, \ell) = (y^*, \ell^*))$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau} \cdot V^{\tau v_{y,\ell,b}}_{\ell,b}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}^{v_{y,\ell,b}}_{\ell,b}$

In words, we can say that, the ciphertext component $C_{y^*,\ell^*,1-b^*}$ also includes the random elements in the subgroup $\mathbb{G}_p$ whereas in Hyb_1 only C_{y^*,ℓ^*,b^*} for index i^* included a random elements in the subgroup $\mathbb{G}_p$.

$\mathbf{Hyb_3}$: This hybrid consists some sub-hybrids $\mathrm{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ where $(\widetilde{\ell}, \widetilde{b}) \in [k] \times \{0, 1\}$ is the same as Hyb_2 except that the column components in the challenge ciphertext components are as in following Table 18. In words, we can say that the ciphertext components $C_{y^*,\ell,b}$ for

Table 18: Computing column components of the sub-hybrid $\mathrm{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee$ $(y = y^* \wedge \ell \notin [\widetilde{\ell}] \cup \{\ell^*\}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$H^t_{y,\ell,b} h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee$ $(y = y^* \wedge \ell \in [\widetilde{\ell}-1] \cup \{\ell^*\}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b \leq \widetilde{b})$	$H^t_{y,\ell,b} h^{w_{y,\ell,b}\tau} V_{\ell,b}^{\tau v_{y,\ell,b}}$	$g^{w_{y,\ell,b}}$

$\ell < \widetilde{\ell}$, or $\ell = \widetilde{\ell}$ and $b \leq \widetilde{b}$ include a random component in the subgroup $\mathbb{G}_p$.

$\mathbf{Hyb_4}$: This hybrid is similar as the previous sub-hybrid $\mathrm{Hyb}_{3,k,1}$ except that the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^* + 1), \perp, 0)$. Note that, if $y^* = \widetilde{n}$, it also recall that the special-encryption algorithm can be directly extended to encrypt to such position.

$\mathbf{Hyb_5}$. This hybrid corresponds to the upper identity-hiding game in which the challenge ciphertext is a special encryption to index-position-bit tuple $(i^* + 1, \perp, 0)$. Note that, if $y^* \neq \widetilde{n}$ then hybrids 4 and 5 are already identical.

Next, we discuss the indistinguishability of the above hybrids. By combining above consecutive hybrids, the upper identity-hiding security holds.

$\mathbf{Hyb_1} \approx \mathbf{Hyb_2}$: The proof of the indistinguishability of the hybrids Hyb_1 and Hyb_2 is identical to that Claim 6 and Lemma 8.

$\mathbf{Hyb}_{3,\widetilde{\ell},\widetilde{b}} \approx \mathbf{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$: If the modified-2 D3DH assumption 3 holds, then there does not exists any PPT adversary can distinguish between the $\mathrm{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ and $\mathrm{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,\widetilde{b}+1 \mod 2}$ non-negligible advantage.

First note that, $\mathrm{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ is identically equals to $\mathrm{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,\widetilde{b}+1 \mod 2}$ for $\widetilde{\ell} = \ell^*$. Otherwise, according to the key query there have two cases:

Case 1. Adversary makes key query of the index tuple (j, id) with $j = i^* \wedge \mathsf{id}_{\widetilde{\ell}} = \widetilde{b}$.

To prove the indistinguishable in that case, Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ which can distinguish the $\mathrm{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ and the $\mathrm{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ with non-negligible advantage $\epsilon(\cdot)$. We construct a PPT reduction algorithm $\mathscr{B}$ which breaks the modified-2 D3DH assumption 3 with the same non-negligible advantage as follows:

$$\left(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, D = g_p^{b^2}, T\right)$$

where T is either g_p^{abc} or a uniformly random element from the subgroup $\mathbb{G}_p$. Next, $\mathscr{B}$ receives a tuple $(1^\lambda, 1^n, 1^k, 1^m, (i^*, \ell^*, b^*))$ from the adversary $\mathscr{A}$ and then $\mathscr{B}$ simulates the master public key and sends it to the adversary. After seeing mpk, the adversary makes polynomial times key queries for the secret keys of distinct index positions. After simulating the public keys, secret keys and the challenge ciphertext, the adversary finally outputs a bit $\mathfrak{b}'$ as guess which $\mathscr{B}$ uses to break the assumption 3. As the reduction plays the game with its challenger in the subgroup $\mathbb{G}_p$ so any elements it can choose from the subgroup $\mathbb{G}_q$ by itself. Let us implicitly set the exponents as below

$$t_p = ab; \;\; r_{p,x^*,j} = b \cdot \widetilde{r}_{p,x^*,j}; \;\; c_{p,y^*,\ell^*,b^*} = c + \widetilde{c}_{p,y^*,\ell^*,b^*};$$

$$s_{p,x^*} = \widetilde{s}_{p,x^*}/b;\;\; c_{p,y^*,\widetilde{\ell},\widetilde{b}} = -c + \widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}};\;\; \kappa_{p,x^*} = \widetilde{\kappa}_{p,x^*}/ab$$

where $\widetilde{r}_{p,x^*,j}, \widetilde{s}_{p,x^*}, \widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}}, \widetilde{\kappa}_{p,x^*}$ and $\widetilde{c}_{p,y^*,\ell^*,b^*}$ are the random exponents. Also, we implicitly set $h_p = B = g_p^b$. and $f_p = B^{d_1}$ for some exponent $d_1 \leftarrow \mathbb{Z}_N$. Setting the exponents allows to simulate the public key, secret key exactly as well as the challenge group elements T which can be programmed in the challenge ciphertext components $C_{y^*,\widetilde{\ell},\widetilde{b}}$.

Public key simulation. Let us consider two random group generators $h_q, f_q \in \mathbb{G}_q$ by sampling random exponent $d, d' \in \mathbb{Z}_N$ such that $h_q = g_q^d, f_q = g_q^{d'}$. To simulate the public key, $\mathscr{B}$ additionally chooses the following exponents

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, &\quad \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], &\quad \widetilde{r}_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

It samples $\beta \leftarrow \mathbb{Z}_N$ and choose random generator $f_q \leftarrow \mathbb{G}_q$ and sets $f_q = g_q^{d'}$ for some $d' \in \mathbb{Z}_N$. Next, $\mathscr{B}$ computes the master public key's components by using the modified-2 D3DH instances of assumption 3.

$$E_{x,j} = \begin{cases} g^{\widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ (Bg_q)^{\widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases}, \qquad F_{x,j} = \begin{cases} (Bh_q)^{\widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ (Dh_q)^{\widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases},$$

$$H_{y,\ell,b} = \begin{cases} Cg^{\widetilde{c}_{y,\ell,b}} & \text{if } (y,\ell,b) = (y^*,\ell^*,b^*), \\ C^{-1}g^{\widetilde{c}_{y,\ell,b}} & \text{if } (y,\ell,b) = (y^*,\widetilde{\ell},\widetilde{b}), \\ g^{\widetilde{c}_{y,\ell,b}} & \text{elsewhere.} \end{cases} \qquad W_{x,j} = e(Bf_q, Bf_q)^{d_1^2\psi_{x,j}}\ \forall x$$

It also computes $E_{q,x,j} = g_q^{\beta\widetilde{r}_{x,j}}, F_{q,x,j} = h_q^{\beta\widetilde{r}_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, G_{q,x,j} = e(g_q,g_q)^{\beta\alpha_{x,j}}, W_{q,x,j} = e(f_q,f_q)^{\beta\psi_{x,j}}$ and sends the master public key as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = Bh_q, f = B^{d_1} f_q, E_q = g_q^{\beta}, Z_q = f_q^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j}, W_{x,j}, \\ E_{q,x,j}, F_{q,x,j}, G_{q,x,j}, W_{q,x,j} \end{array} \right\}_{x\in[\widehat{n}], j\in[m]} \\ \{H_{y,\ell,b}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}}, \\ \left\{\widetilde{V}_{\ell,b}, V_{\ell,b}\right\}_{(\ell,b)\in[k]\times\{0,1\}} \end{array} \right)$$

Secret key simulation. $\mathscr{B}$ answers the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, K_1, K_2)$ query for the tuple $(i = (x,y), \mathsf{id}, \boldsymbol{u})$ as follows:

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot g^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x \neq x^*, y \neq y^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (Bg_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x = x^*, y \neq y^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (\prod_{\ell\in[k]} H_{y,\ell,\mathsf{id}_\ell})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle} & \text{if } x \neq x^*, y = y^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (Bg_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } i = i^*, \mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b}, \mathsf{id}_{\ell^*} \neq b^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (Bg_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } i = i^*, \mathsf{id}_{\widetilde{\ell}} = \widetilde{b}, \mathsf{id}_{\ell^*} = b^*. \end{cases}$$

$$K_2 = (B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \text{ for all } x.$$

There have some restrictions over the key queries to the key generation oracle i.e.,

– Adversary $\mathscr{A}$ can not query for the tuple $(i, \mathsf{id}, \boldsymbol{u})$ to the key generation oracle such that $i = i^* \wedge \mathsf{id}_{\ell^*} \neq b^*$.
– In case-1 adversary also makes an query for the tuple $(i, \mathsf{id}, \boldsymbol{u})$ where $i = i^* \wedge \mathsf{id}_{\widetilde{\ell}} = \widetilde{b}$. If an adversary makes a key query for the challenge index-identity-vector tuple $i = i^* \wedge \mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b}$, then challenger strictly aborts the upper identity-hiding experiment.

Ciphertext simulation. To generate the ciphertext, the challenger chooses the exponents as follows.

$$\begin{aligned} \tau \in \mathbb{Z}_N,\ \ t_q \leftarrow \mathbb{Z}_N,\ \ \forall j \in [m],\ \ \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N, \\ \forall x \in [\widehat{n}],\ \ e_x, f_x, d_x, \widetilde{\kappa}_x, \widetilde{s}_x \leftarrow \mathbb{Z}_N, \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\},\ \ \widetilde{w}_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N. \end{aligned}$$

Now, the challenger computes the row and column components as the tables below.

Table 19: Computing row components of the ciphertext for $x \in [\widehat{n}], j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{\widetilde{s}_x}$	$F_{q,x,j}^{\widetilde{s}_x \tau}$	$E_q^{\widetilde{s}_x t_q}$	$Z_q^{\widetilde{\kappa}_x t_q}$	$e(g_q, g_q)^{\nu_j} \cdot G_{q,x,j}^{\widetilde{s}_x t_q} \cdot W_{q,x,j}^{\widetilde{\kappa}_x t_q}$
$x = x^*$	$g^{\widetilde{s}_x \widetilde{r}_{x,j}}$	$(Bh_q)^{\widetilde{s}_x \widetilde{r}_{x,j} \tau}$	$(Ag_q^{t_q})^{\widetilde{s}_x}$	$(B^{d_1} f_q^{t_q})^{\widetilde{\kappa}_x}$	$e(g,g)^{\nu_j} \cdot e(g, Ag_q^{t_q})^{\widetilde{s}_x \alpha_{x,j}} \cdot$ $e(B,B)^{d_1^2 \psi_{x,j} \widetilde{\kappa}_x} \cdot e(f_q, f_q)^{\psi_{x,j} \widetilde{\kappa}_x t_q}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$h^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	$(B^{d_1} f_q)^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 20: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell > \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell \notin [\widetilde{\ell}] \cup \{\ell^*\}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\widetilde{w}_{y,\ell,b} \tau}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} g^{\widetilde{w}_{y,\ell,b}}$
$y = y^* \wedge \ell = \widetilde{\ell} \wedge b = \widetilde{b}$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot T^{-1}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} \cdot g^{\widetilde{w}_{y,\ell,b}}$
$(y < y^*) \vee (y = y^* \wedge \ell < \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell \in [\widetilde{\ell} - 1] \cup \{\ell^*\}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot g_p^{\upsilon_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$

After seeing the challenge ciphertext $\mathscr{B}$ receives a guess bit $\mathfrak{b}'$ from $\mathscr{A}$ and $\mathscr{B}$ forwards it to the modified-2 D3DH challenger. If $T = g_p^{abc}$, then $\mathscr{B}$ simulates the view of $\mathsf{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ otherwise, if T is uniformly choose a group element from the subgroup $\mathbb{G}_p$ then $\mathscr{B}$ simulates the view of $\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$. Thus, if the adversary $\mathscr{A}$ wins with the advantage $\epsilon(\cdot)$ then $\mathscr{B}$ breaks the assumption 3 with same advantage.

Case 2. Adversary makes key queries for the index j and identity id such that $(j \neq i^* \vee \mathsf{id}_{\widetilde{\ell}} = 1 - \widetilde{b})$.

Since in this case, the adversary is allowed to secret key query for the index j such that $j \neq i^*$ and an identity id satisfies the conditions $\mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b} \wedge \mathsf{id}_{\ell^*} = b^*$. So, we can use the same proof strategy as used in Claim 6 and 8 where the reduction algorithm does not need to know the value of the group element $g^{\langle \boldsymbol{r}_{x^*}, \boldsymbol{u} \rangle c_{y^* m \widetilde{\ell}, \widetilde{b}}}$ for answering the key queries. The proof technique is similar with the proof of Claim 6 and Lemma 8.

$\mathbf{Hyb_3} \approx \mathbf{Hyb_4}$: Since, the hybrid 3 and hybrid 4 hold directly. Therefore, no PPT adversary can distinguish between these two hybrids with non-negligible advantage.

$\mathbf{Hyb}_4 \approx \mathbf{Hyb}_5$: The indistinguishable of the Hyb_4 and Hyb_5 can be classified into two cases.

Case 1. If $y^* \neq \widetilde{n}$, then both the Hyb_4 and Hyb_5 are identical.

Case 2. For $y^* = \widetilde{n}$, then the indistinguishability of hybrid 4 and hybrid 5 is followed from a sequence of hybrid games which is similar as the claim 7. In particular, the hybrid Hyb_4 is similar to the hybrid 2 and Hyb_5 is identical with the hybrid 7 as described in the claim 7. Thus, this indistinguishability follows from the claim 7.

This concludes the upper identity-hiding security game. □

Lemma 10 (Message-hiding) *Assuming the joint-D3DH 7 holds over the bilinear group $\mathbb{BG}$, the our EIPL-IPFE satisfies message-hiding security as per the Definition 9.*

Proof. The message-hiding security requires that the special encryptions of $\boldsymbol{v}^{(0)}$ and $\boldsymbol{v}^{(1)}$ to the same tuple $(i^*, \perp, 0)$ are indistinguishable if all the secret key queries of the form $(i \geq i^*, \mathsf{id}, \boldsymbol{u})$ must satisfy $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$. To prove of the above Lemma 10, we consider the following hybrid games.

Hybrid 1. The hybrid corresponds to the message-hiding game in which the challenge ciphertext is a special encryption of the vector $\boldsymbol{v}^{(\mathfrak{b})}$ to the index-position-value tuple $(i^*, \perp, 0)$.

Hybrid 2. This hybrid is similar to the hybrid 1 except that the row components $I_{x,j}$ for $x \geq x^*$ as mentioned in the following Table 21.

Table 21: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t_q}$	$Z_q^{\kappa_x t}$	$e(g_q, g_q)^{v_j^{(\mathfrak{b})}} \cdot e(g_q, A_x)^{\alpha_{x,j}} \cdot e(f_q, B_x)^{\psi_{x,j}}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$g^{s_x t}$	$f^{\kappa_x t}$	$e(g, g)^{v_j^{(\mathfrak{b})}} \cdot e(g, A_x)^{\alpha_{x,j}} \cdot e(f, B_x)^{\psi_{x,j}}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$h^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	f^{d_x}	$e(g, g)^{f_x \phi_j}$

Hybrid 3. Hybrid 3 is same as hybrid 2 except that the row components B_x for $x \geq x^*$ as mentioned in the Table 22 below.

Table 22: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t_q}$	$Z_q^{\kappa_x (t+t')}$	$e(g_q, g_q)^{v_j^{(\mathfrak{b})}} \cdot e(g_q, A_x)^{\alpha_{x,j}} \cdot e(f_q, B_x)^{\psi_{x,j}}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$g^{s_x t}$	$f^{\kappa_x (t+t')}$	$e(g, g)^{v_j^{(\mathfrak{b})}} \cdot e(g, A_x)^{\alpha_{x,j}} \cdot e(f, B_x)^{\psi_{x,j}}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$h^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	f^{d_x}	$e(g, g)^{f_x \phi_j}$

where t' is uniformly chosen from $\mathbb{Z}_N$. From hybrid 3, the adversary can not extract any information about the challenge bit $\mathfrak{b}$ from the row component $I_{x,j}$ for all $x \in [\widehat{n}]$, $j \in [m]$.

Next, we discuss the indistinguishability of the above hybrids.

Hybrid 1 $\approx$ Hybrid 2: From the hybrid 1, we modify the challenge ciphertext generation process by using the master secret key $\{(r_{x,j}, \alpha_{x,j}, \psi_{x,j})\}_{x \in [\widehat{n}], j \in [m]}$. First, the challenger $\mathscr{B}$ computes all the ciphertext components as

$$G_{q,x,j} = e(g_q, g_q)^{\beta s_x t \alpha_{x,j}} \quad G_{x,j} = e(g,g)^{s_x t \alpha_{x,j}}$$
$$W_{q,x,j} = e(f_q, f_q)^{\beta \kappa_x t \psi_{x,j}} \quad W_{x,j} = e(f,f)^{\kappa_x t \psi_{x,j}}$$

for a randomly chosen $\beta, s_x, \kappa_x, t \leftarrow \mathbb{Z}_N$. For $x > x^*$, $\mathscr{B}$ uses the master secret keys to compute $I_{x^*,j}$ as below:

$$\begin{aligned} I_{x,j} &= e(g_q, g_q)^{\nu_j^{(\mathfrak{b})}} \cdot e(g_q, g_q)^{\beta s_x t \alpha_{x,j}} \cdot e(f_q, f_q)^{\beta \kappa_x t \psi_{x,j}} \\ &= e(g_q, g_q)^{\nu_j^{(\mathfrak{b})}} \cdot e(g_q, g_q^{\beta s_x t})^{\alpha_{x,j}} \cdot e(f_q, f_q^{\beta \kappa_x t})^{\psi_{x,j}} \\ &= e(g_q, g_q)^{\nu_j^{(\mathfrak{b})}} \cdot e(g_q, A_x)^{\alpha_{x,j}} \cdot e(f_q, B_x)^{\psi_{x,j}} \end{aligned}$$

Also, for $x = x^*$

$$\begin{aligned} I_{x,j} &= e(g,g)^{\nu_j^{(\mathfrak{b})}} \cdot e(g,g)^{s_x t \alpha_{x,j}} \cdot e(f,f)^{\kappa_x t \psi_{x,j}} \\ &= e(g,g)^{\nu_j^{(\mathfrak{b})}} \cdot e(g, g^{s_x t})^{\alpha_{x,j}} \cdot e(f, f^{\beta \kappa_x t})^{\psi_{x,j}} \\ &= e(g,g)^{\nu_j^{(\mathfrak{b})}} \cdot e(g, A_x)^{\alpha_{x,j}} \cdot e(f, B_x)^{\psi_{x,j}} \end{aligned}$$

It is clearly observed that the above distribution of challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\mathfrak{b})}} = (\{R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}\}_{x \in [\widehat{n}], j \in [m]}, \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}})$ remains unaltered in the hybrid 2. So we can argue that these two hybrids 1 and 2 are indistinguishable.

Hybrid 2 ≈ Hybrid 3: In this hybrid, we modify the the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\mathfrak{b})}} = (\{R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}\}_{x \in [\widehat{n}], j \in [m]}, \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}})$ as

$$B_x = f_q^{\beta \kappa_x (t+t')} \quad \text{for } x > x^*$$

where $t' \leftarrow \mathbb{Z}_N$. Other remaining ciphertext components are similarly generated as hybrid 1. In the following, it will be shown that if the joint-D3DH assumption 7 holds over the bilinear group $\mathbb{BG}$, then this modification should not significantly affect the adversary's view.

Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ that distinguishes between the hybrid 2 and 3 with non-negligible advantages. We construct a PPT reduction algorithm which breaks the joint-D3DH assumption 7 with same non-negligible advantages as follows.

The reduction algorithm $\mathscr{B}$ first receives the challenges of joint-D3DH assumption 7 from the challenger as

$$\left(\mathbb{BG}, g_p, g_q, A_1 = g_q^a, A_2 = g_p^a, B_1 = g_q^b, B_2 = g_p^b, C_1 = g_q^c, C_2 = g_p^c, (T,S)\right)$$

where (T,S) is either (g_q^{abc}, g_p^{abc}) or a random element $(g_q^{t+t'}, g_p^{t+t'})$ from $\mathbb{G}_q \times \mathbb{G}_p$ with $t' \leftarrow \mathbb{Z}_N$. Next, the challenger receives the challenge tuple $(1^\lambda, n, 1^k, 1^m, i^* = (x^*, y^*))$ from the adversary $\mathscr{A}$. We apply the reduction game with its challenger in the subgroup $\mathbb{G}_q$ for $x \geq x^*$. Sets $f_q = g_q^{d_q}, f_p = g_p^{d_p}$ where d_p, d_q are sampled randomly from $\mathbb{Z}_N$. We also implicitly set the exponents $d_q = d_p = a, t_q = t_p = c$ and $\kappa_{q,x} = b \cdot \widetilde{\kappa}_{q,x}, \kappa_{p,x} = b \cdot \widetilde{\kappa}_{p,x}$ where $\widetilde{\kappa}_{q,x}, \widetilde{\kappa}_{p,x}$ are chosen at random for $x \geq x^*$. Setting exponents this way, it allows us to simulate the master public key, secret key and challenge ciphertext as well as the group elements T, S can be programmed in the challenge ciphertext component $I_{x,j}$ for $x \geq x^*$.

Public key simulation. To simulate the public key, first chose random exponents as follows:

$$\forall \ell \in [k], b \in \{0,1\}, \quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p$$
$$\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad c_{y,\ell,b} \leftarrow \mathbb{Z}_N$$

$$\forall j \in [m], x \in [\widehat{n}], \quad r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N$$

It samples $\beta \leftarrow \mathbb{Z}_N$. Next, it computes all public key components are given below.

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, h, g = g_p g_q, f = A_1 A_2, E_q = g_q^{\beta}, Z_q = A_1^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j} = g^{r_{x,j}}, F_{x,j} = h^{r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = g_q^{\beta r_{x,j}}, F_{q,x,j} = h_q^{\beta r_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}}, \\ W_{q,x,j} = e(A_1, A_2)^{\beta \psi_{x,j}}, W_{x,j} = e(A_1 A_2, A_1 A_2)^{\psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b} = g^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

Note that, all the public key terms should be computed using the joint-D3DH instances of assumption 7.

Secret key simulation. The challenger $\mathscr{B}$ answers the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, K_1, K_2)$ corresponding to the tuple $(i = (x, y), \mathsf{id}, \boldsymbol{u})$ where K_1, K_2 is computed as follows

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle} \quad \text{if } (x \geq x^*) \vee (x = x^* \wedge y \geq y^*),$$
$$K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} = (A_1 \cdot A_2)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}.$$

Note that, the adversary can key queries for the index i such that $i \geq i^*$ whenever $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$.

Ciphertext simulation. The challenger $\mathscr{B}$ chooses the random exponents as follows:

$$\forall j \in [m], \ \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N, \ \tau, t_p \leftarrow \mathbb{Z}_N,$$
$$\forall x \in [\widehat{n}], \quad e_x, f_x, s_x, \widetilde{\kappa}_x \leftarrow \mathbb{Z}_N,$$
$$\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad w_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N.$$

Next, challenger $\mathscr{B}$ uses the joint-D3DH assumption 7 challenge instances to simulate the challenge ciphertext, which mention in Tables 23 and 24.

Table 23: Computing row components of the ciphertext for $x \in [\widehat{n}], j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$C_1^{s_x}$	$T^{\beta \widetilde{\kappa}_x}$	$e(g_q, g_q)^{\nu_j^{(\mathsf{b})}} \cdot e(g_q, A_x)^{\alpha_{x,j}} \cdot e(A_1, B_x)^{\psi_{x,j}}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$(C_1 C_2)^{s_x}$	$(ST)^{\widetilde{\kappa}_x}$	$e(g,g)^{\nu_j^{(\mathsf{b})}} \cdot e(g, A_x)^{\alpha_{x,j}} \cdot e(A_1 A_2, B_x)^{\psi_{x,j}}$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	$(A_1 A_2)^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 24: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge (\ell, b^*) \neq (\ell^*, b^*))$	$(C_1 C_2)^{c_{y,\ell,b}} \cdot h^{w_{y,\ell,b} \tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee ((y, \ell, b) = (y^*, \ell^*, b^*))$	$(C_1 C_2)^{c_{y,\ell,b}} \cdot h^{w_{y,\ell,b} \tau} \cdot V_{\ell,b}^{\tau \upsilon_{y,\ell,b}}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{\upsilon_{y,\ell,b}}$

If $(T, S) = (g_q^{abc}, g_p^{abc})$, $\mathscr{B}$ simulates the view of hybrid 2, otherwise if $(T, S) = (g_q^{t+t'}, g_p^{t+t'})$

for random $t' \leftarrow \mathbb{Z}_N$, then $\mathscr{B}$ simulates the view of hybrid 3. Therefore, if $\mathscr{A}$ wins with the advantage $\epsilon(\cdot)$, then $\mathscr{B}$ breaks the joint-D3DH assumption 7 with same advantage.

In this hybrid, we claim that the challenge ciphertext $\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}$ perfectly hides the bit $\mathfrak{b} \in \{0,1\}$. Recall that, for all $x > x^*$, we have

$$\begin{aligned}
I_{x,j} &= e(g_q, g_q)^{v_j^{(\mathfrak{b})}} \cdot e(g_q, A_x)^{\alpha_{x,j}} \cdot e(f_q, B_x)^{\psi_{x,j}} \\
&= e(g_q, g_q)^{v_j^{(\mathfrak{b})}} \cdot e(g_q, E_q^{\beta s_x t})^{\alpha_{x,j}} \cdot e(f_q, Z_q^{\kappa_x \beta (t+t')})^{\psi_{x,j}} \\
&= e(g_q, g_q)^{v_j^{(\mathfrak{b})}} \cdot e(g_q, g_q)^{\beta s_x t \alpha_{x,j}} \cdot e(g_q, g_q)^{d_q^2 \kappa_x \beta (t+t') \psi_{x,j}} \; [\text{as } f_q = g_q^{d_q}] \\
&= e(g_q, g_q)^{v_j^{(\mathfrak{b})} + d_q^2 \kappa_x \beta t' \psi_{x,j}} \cdot e(g_q, g_q)^{\beta s_x t \alpha_{x,j}} \cdot e(g_q, g_q)^{d_q^2 \kappa_x \beta t \psi_{x,j}}
\end{aligned}$$

For all $j \in [m]$, we consider

$$z_j^{(\mathfrak{b})} = v_j^{(\mathfrak{b})} + d_q^2 \kappa_x \beta t' \psi_{x,j} \implies \boldsymbol{z}^{(\mathfrak{b})} = \boldsymbol{v}^{(\mathfrak{b})} + d_q^2 \kappa_x \beta t' \boldsymbol{\psi}_x \in \mathbb{Z}_N^m$$

To prove that $\boldsymbol{z}^{(\mathfrak{b})}$ does not reveal any information about $\mathfrak{b} \in \{0,1\}$ to any legitimate adversary, we consider $\boldsymbol{v} = \boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)} \mod N$ and generates a $(m-1)$ dimensional subspace with its $\mathbb{Z}_q$ basis $\mathbf{V} \in \mathbb{Z}_N^{(m-1)\times m}$ as

$$\begin{aligned}
\mathsf{orth}(\boldsymbol{v}) &= \{\boldsymbol{u} \in \mathbb{Z}_N^m : \langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0 \mod N\} \\
&= \{\boldsymbol{u} \in \mathbb{Z}_N^m : \langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle \mod N\}
\end{aligned}$$

Let choose a vector $\widetilde{\boldsymbol{u}} \notin \mathsf{orth}(\boldsymbol{v})$ in a deterministic manner and set a $m \times m$ invertible matrix $\widetilde{\mathbf{V}}$ as

$$\widetilde{\mathbf{V}} = \begin{bmatrix} \mathbf{V} \\ \widetilde{\boldsymbol{u}}^\top \end{bmatrix}$$

To prove, $\boldsymbol{z}^{(\mathfrak{b})}$ does not reveal any information about the challenge bit $\mathfrak{b} \in \{0,1\}$, it is suffices to prove that $\widetilde{\mathbf{V}} \cdot \boldsymbol{z}^{(\mathfrak{b})}$ information-theoretically hides the bit $\mathfrak{b}$. As $\mathbf{V} \cdot \boldsymbol{v}^{(0)} = \mathbf{V} \cdot \boldsymbol{v}^{(1)}$, the first $(m-1)$ rows of $\widetilde{\mathbf{V}} \cdot \boldsymbol{z}^{(\mathfrak{b})}$ are clearly independent of $\mathfrak{b}$. We now concentrate on the last row of the product $\widetilde{\mathbf{V}} \cdot \boldsymbol{z}^{(\mathfrak{b})}$ i.e.,

$$\langle \boldsymbol{v}^{(\mathfrak{b})}, \widetilde{\boldsymbol{u}} \rangle + \beta d_q^2 t' \kappa_x \cdot \langle \boldsymbol{\psi}_x, \widetilde{\boldsymbol{u}} \rangle \tag{14}$$

We have to show that from the Equation 14, adversary can not learn any information about the challenge bit $\mathfrak{b}$.

Let $(\boldsymbol{r}_x^0, \boldsymbol{\alpha}_x^0, \boldsymbol{\psi}_x^0) \in \mathbb{Z}_N^m \times \mathbb{Z}_N^m \times \mathbb{Z}_N^m$ denotes a tuple of vectors satisfying the relation

$$\left\{ G_{q,x,j}^{s_x t} \cdot W_{q,x,j}^{\kappa_x t} = e(g_q, g_q)^{\beta s_x t \alpha_{x,j}^0} \cdot e(f_q, f_q)^{\beta \kappa_x t \psi_{x,j}^0} \right\}_{j \in [m]}$$

and the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, K_1, K_2)$ associated with a tuple $(i, \mathsf{id}, \boldsymbol{u})$ are given below

$$K_1 = g^{\langle \boldsymbol{\alpha}_x^0, \boldsymbol{u} \rangle} \cdot \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\langle \boldsymbol{r}_x^0, \boldsymbol{u} \rangle}, \quad K_2 = f^{\langle \boldsymbol{\psi}_x^0, \boldsymbol{u} \rangle}$$

Since all the key queries vector $\boldsymbol{u} \in \mathsf{orth}(\boldsymbol{v})$, the joint distribution of the secret key is

$$\{\boldsymbol{r}_x^0 + \mu \cdot \boldsymbol{v}, \boldsymbol{\alpha}_x^0 - \mu \cdot \iota \cdot \boldsymbol{v} \mod N, \boldsymbol{\psi}_x^0 + \mu \cdot \boldsymbol{v} \mod N : \mu \in \mathbb{Z}_N\}$$

where $\iota = \log_{e(g_q, g_q)} e(f_q, f_q)$. Therefore, the conditional distribution of $\beta d_q^2 t' \kappa_x \cdot \langle \boldsymbol{\psi}_x, \widetilde{\boldsymbol{u}} \rangle \mod N$ becomes

$$\{\beta d_q^2 t' \kappa_x \cdot \langle \boldsymbol{\psi}_x^0 + \mu \cdot \boldsymbol{v}, \widetilde{\boldsymbol{u}} \rangle \mod N : \mu \in \mathbb{Z}_N\}$$

$$= \{\beta d_q^2 t' \kappa_x \cdot (\langle \boldsymbol{\psi}_x^0, \widetilde{\boldsymbol{u}} \rangle + \mu \langle \boldsymbol{v}, \widetilde{\boldsymbol{u}} \rangle : \mu \in \mathbb{Z}_N)\} \tag{15}$$

Since, $\widetilde{\boldsymbol{u}} \notin \mathsf{orth}(\boldsymbol{v})$, so $\langle \boldsymbol{v}, \widetilde{\boldsymbol{u}} \rangle \neq 0$ and $t' \neq 0$ with a high probability. So from the Equation 15, it is uniformly distributed over $\mathbb{Z}_N$. Therefore, the term $\beta d_q^2 t' \kappa_x \cdot \langle \boldsymbol{\psi}_x, \widetilde{\boldsymbol{u}} \rangle \mod N$ information theoretically hides the inner product $\langle \boldsymbol{v}^{(\mathfrak{b})}, \boldsymbol{u} \rangle$ in the inner product $\langle \boldsymbol{z}^{(\mathfrak{b})}, \boldsymbol{u} \rangle \mod N$. From the component $B_x = f_q^{\beta \kappa_x (t+t')}$ if $x > x^*$, the adversary can not extract the information about $(t+t')$. So information theoretically $\beta \kappa_x (t+t')$ hides $(t+t')$ as κ_x is uniformly chosen from $\mathbb{Z}_N$.

By the similar argument for $x = x^*$, we can say that the challenge ciphertext hides the information about $\mathfrak{b}$.

This concludes the proof of message-hiding. □

7 Selectively secure EIPL-IBIPFE using Bilinear Maps

Let us assume $\mathcal{G}_{\mathsf{BG.Gen}}$ be a bilinear group generator of a composite order group $N = p \cdot q$ where p, q be two prime integers. Now, we describe our EIPL-IBIPFE = (Setup, KeyGen, Enc, SplEnc, Dec) scheme using bilinear map.

Setup$(1^\lambda, n, 1^k, 1^{k'}, 1^m) \to$ (msk, mpk, key): A trusted authority takes a security parameter λ, the user identity space parameter k, group identity space parameter k', index space parameter n and message vector length m as input and executes the algorithm as follows:

- sets $\widetilde{n} = \lceil \sqrt{\frac{n}{k}} \rceil$ and $\widehat{n} = \lceil \frac{n}{\widetilde{n}} \rceil$.
- samples a bilinear group $\mathbb{BG} = (p, q, N = pq, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot)) \leftarrow \mathcal{G}_{\mathsf{BG.Gen}}(1^\lambda)$.
- chooses random generator $g_p, h_p, f_p \in \mathbb{G}_p$ and $g_q, h_q, f_q \in \mathbb{G}_q$ and sets $g = g_p g_q, h = h_p h_q$, $f = f_p f_q \in \mathbb{G}$. Also, it randomly chooses some exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, \quad & \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\} \quad & c_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}] \quad & r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

- samples $\vartheta'_p \leftarrow \mathbb{G}_p, \vartheta'_q \leftarrow \mathbb{G}_q$ such that $\vartheta' = \vartheta'_p \vartheta'_q \in \mathbb{G}$ and a k'-length vector $\boldsymbol{\vartheta} = (\vartheta_i)_{i \in [k']} = (\vartheta_{p,i} \vartheta_{q,i})_{i \in [k']}$ whose each $\vartheta_{p,i}, \vartheta_{q,i}$ are chosen at random from the subgroups $\mathbb{G}_p, \mathbb{G}_q$ respectively. Let gid be a k'-bit string representing a group identity, where gid_i denotes the i-th bit of gid and $\mathcal{V} \subseteq \{1, 2, \ldots, k'\}$ be set of all i for which $\mathsf{gid}_i = 1$. Consider two identity encoding functions H, H_q be defined as $\mathrm{H}(\mathsf{gid}) = (\vartheta'_p \vartheta'_q) \prod_{i \in \mathcal{V}} \vartheta_{p,i} \vartheta_{q,i}$ and $\mathrm{H}_q(\mathsf{gid}) = \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{q,i}$ for $\mathsf{gid} \in \mathcal{GID}$.
- chooses $\beta, \widehat{r} \leftarrow \mathbb{Z}_q$. and sets

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, h, g, f, \vartheta', \vartheta_q'^{\beta}, \boldsymbol{\vartheta}, \{\vartheta_{q,i}^{\beta}\}_{i \in [k']}, \mathrm{H}, \mathrm{H}_q, E_q = g_q^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j} = g^{\widehat{r} r_{x,j}}, F_{x,j} = h^{\widehat{r} r_{x,j}}, Y_{x,j} = g^{\psi_{x,j}}, \\ E_{q,x,j} = g_q^{\beta \widehat{r} r_{x,j}}, F_{q,x,j} = h_q^{\beta \widehat{r} r_{x,j}}, Y_{q,x,j} = g_q^{\beta \psi_{x,j}}, \\ G_{x,j} = e(g,g)^{\alpha_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b} = g^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

$$\mathsf{msk} = \begin{pmatrix} \mathbb{G}, g, \widehat{r}, \{r_{x,j}, \alpha_{x,j}, \psi_{x,j}\}_{x\in[\widehat{n}], j\in[m]}, \\ \{c_{y,\ell,b}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}} \end{pmatrix}, \quad \mathsf{key} = \mathsf{mpk}.$$

– Finally, it publishes the master public key mpk and keeps the master secret key msk secret to itself.

KeyGen(msk, i, id, gid, $\boldsymbol{u}$) → $\mathsf{sk}_{\boldsymbol{u}}$: On input the master secret key msk, an index $i \in [n]$, an user identity $\mathsf{id} \in \{0,1\}^k$, an group identity $\mathsf{gid} \in \{0,1\}^{k'}$, and the input vector $\boldsymbol{u} \in \mathbb{Z}^m$, the trusted authority proceeds as follows:

– consider $(x, y) \in [\widehat{n}] \times [\widetilde{n}]$ be the unique row wise representation of index i (for any $i \in [n]$, its corresponding indices can be defined as $y = i \mod \widetilde{n}$ and $x = \lceil \frac{i}{\widetilde{n}} \rceil$).
– chooses a random $\widetilde{r} \leftarrow \mathbb{Z}_N$ and sets $r = \widetilde{r} \cdot \widehat{r}$.
– computes $\mathrm{H}(\mathsf{gid}) = \vartheta' \prod_{i\in\mathcal{V}} \vartheta_i$ where $\mathcal{V} = \{i : i\text{-th entry of gid is equals to } 1\}$.
– outputs the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, \mathsf{gid}, K = (K_1, K_2, K_3))$ where

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \left(\prod_{\ell\in[k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle}, \quad K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \cdot \mathrm{H}(\mathsf{gid})^r, \quad K_3 = g^r$$

Enc(mpk, gid′, $\boldsymbol{v}$) → $\mathsf{ct}_{\boldsymbol{v}}$: The encryption algorithm is the same as special encryption algorithm (described below) when run on index-position-bit tuple $(i^*, \ell^*, b^*) = (1, \perp, 0)$.

SplEnc(key, gid′, $\boldsymbol{v}$, (i^*, ℓ^*, b^*)) → $\mathsf{ct}_{\boldsymbol{v}}$: On input a key key, a group identity gid′, a message vector $\boldsymbol{v} \in \mathbb{Z}^m$ and index-position-bit tuple (i^*, ℓ^*, b^*), the encryptor performs the algorithm as follows:

– let $(x^*, y^*) \in [\widehat{n}] \times [\widetilde{n}]$ be the unique row-wise representation of the index i^*.
– choose random exponents as

$$\forall\, j \in [m],\ \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N, \quad \tau, t \in \mathbb{Z}_N, \quad \forall\, x \in [\widehat{n}], \quad s_x, e_x, f_x, d_x \leftarrow \mathbb{Z}_N$$
$$\forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad w_{y,\ell,b}, v_{y,\ell,b} \leftarrow \mathbb{Z}_N$$

– for all $x \in [\widehat{n}], j \in [m], (y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$, it generates the following components as described in the Table 25 and Table 26.

Table 25: Computing the row components for $x \in [\widehat{n}], j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x\tau}$	$E_q^{s_x t}$	$\mathrm{H}_q(\mathsf{gid}')^{\beta s_x t}$	$e(g_q, g_q)^{\nu_j} \cdot G_{q,x,j}^{t s_x} \cdot e(f_q, Y_{q,x,j})^{t s_x}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x\tau}$	$g^{s_x t}$	$\mathrm{H}(\mathsf{gid}')^{s_x t}$	$e(g, g)^{\nu_j} \cdot G_{x,j}^{t s_x} \cdot e(f, Y_{x,j})^{t s_x}$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	$\mathrm{H}(\mathsf{gid}')^{d_x}$	$e(g, g)^{f_x \phi_j} \cdot e(f, f)^{f_x \phi_j}$

Table 26: Computing column components for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee (y, \ell, b) = (y^*, \ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau} \cdot V_{\ell,b}^{v_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$

– outputs the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$ associated to the vector $\boldsymbol{v}$ as

$$\mathsf{ct}_{\boldsymbol{v}} = \begin{pmatrix} \{R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}\}_{x\in[\widehat{n}], j\in[m]}, \\ \{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}} \end{pmatrix}$$

Dec($\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}$) $\rightarrow \zeta / \perp$: The decryptor uses the secret key $\mathsf{sk}_{\boldsymbol{u}}$ to decrypts the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$. It computes

$$\eta = \frac{\prod_{j\in[m]} I_{x,j}{}^{u_j} \cdot \prod_{j\in[m]} e(R_{x,j}, \prod_{\ell\in[k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j})}{\prod_{j\in[m]} e(\widetilde{R}_{x,j}, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}) \cdot e(K_1, A_x) \cdot e(K_2, A_x) \cdot e(K_3, B_x)}$$

Finally, it returns $\log\eta$.

7.1 Correctness

In this subsection shows the correctness of our EIPL-IBIPFE from the bilinear map, which is discussed in Sec. 7.

Theorem 8 (Correctness) *If all the components are generated as above algorithms, then our EIPL-IBIPFE scheme is correct with non-negligible probability.*

Proof. Consider the secret key $\mathsf{sk}_{\boldsymbol{u}} = (x, y, \mathsf{id}, \mathsf{gid}, K)$ corresponding to the index $i = (x, y)$, an user identity id, a group gid, and a predicate vector $\boldsymbol{u}$. We know that

$$K = \left(K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot (\prod_{\ell\in[k]} H_{y,\ell,\mathsf{id}_\ell})^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle}, K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid})^r, K_3 = g^r \right)$$

Here, the ciphertext $\mathsf{ct}_{\boldsymbol{v}}$, which is an encryption of a vector $\boldsymbol{v}$ and index-position-value tuple (i^*, ℓ^*, b^*). It consists of $\{R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}\}_{x,j}$, $\{C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}\}_{y,\ell,b}$. Let $i^* = (x^*, y^*)$. From the definition of EIPL-IBIPFE, correctness holds or the decryption oracle gives the outputs $\langle \boldsymbol{u}, \boldsymbol{v}\rangle$ if $\mathsf{gid} = \mathsf{gid}'$ and $(i \geq i^* + 1) \vee ((i^*, \ell^*) = (i, \perp)) \vee ((i^*, \mathsf{id}_{\ell^*}) = (i, 1 - b^*))$. Consider the index position i, an user identity id and the group identity gid which satisfies the above mention constraints then from the representation of i, we can consider the following cases:

Case 1: $x > x^*$. In this case, we have all row components for all $x \in [\widetilde{n}], j \in [m]$ as $R_{x,j} = E_{q,x,j}^{s_x}, \widetilde{R}_{x,j} = F_{q,x,j}^{s_x \tau}, A_x = E_q^{s_x t}, B_x = \mathrm{H}_q(\mathsf{gid})^{\beta s_x t}, I_{x,j} = e(g_q, g_q)^{v_j} \cdot G_{q,x,j}^{t s_x} \cdot e(f_q, Y_{q,x,j})^{t s_x}$. The decryption does not depend whether $y > y^*$ or not, we can compute the following components from the Table 25 and Table 26.

First, we consider $(y > y^*) \vee ((y = y^*) \wedge (\ell, b) \neq (\ell^*, b^*))$ and simplify the following components.

$$\begin{aligned} &\prod_{j\in[m]} e\left(R_{x,j}, \prod_{\ell\in[k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j\in[m]} e\left(g_q^{\beta \widehat{r} r_{x,j} s_x}, \prod_{\ell\in[k]} g^{c_{y,\ell,\mathsf{id}_\ell} t u_j} h^{w_{y,\ell,\mathsf{id}_\ell} \tau u_j}\right) \\ &= e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot e(g_q, h_q)^{\beta \widehat{r} s_x \tau \langle \boldsymbol{u}, \boldsymbol{r}_x\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell}} \end{aligned} \tag{16}$$

$$\begin{aligned} &\prod_{j\in[m]} I_{x,j}{}^{u_j} = e(g_q, g_q)^{\sum_{j\in[m]} u_j v_j} \cdot e(g_q, g_q)^{\sum_{j\in[m]} \beta t s_x \alpha_{x,j} u_j} \cdot e(f_q, g_q)^{\sum_{j\in[m]} \beta \psi_{x,j} t s_x u_j} \\ &= e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v}\rangle} \cdot e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \cdot e(f_q, g_q)^{\beta t s_x \langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \end{aligned} \tag{17}$$

$$\begin{aligned} &\prod_{j\in[m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j\in[m]} e\left(h_q^{\beta \widehat{r} r_{x,j} s_x \tau}, \prod_{\ell\in[k]} g^{w_{y,\ell,\mathsf{id}_\ell} u_j}\right) \\ &= e(h_q, g_q)^{\beta \widehat{r} s_x \tau \langle \boldsymbol{r}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell}} \end{aligned} \tag{18}$$

Also, we have

$$e(K_1, A_x) = e(g_q, g_q)^{\beta s_x t \langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \tag{19}$$

$$e(K_2, A_x) = e(f_q, g_q)^{\beta s_x t \langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot e(\mathrm{H}_q(\mathsf{gid}), g_q)^{\beta s_x t r} \tag{20}$$

$$e(K_3, B_x) = e(g^r, \mathrm{H}_q(\mathsf{gid})^{\beta s_x t}) = e(g_q, \mathrm{H}_q(\mathsf{gid}))^{\beta s_x t r} \tag{21}$$

Therefore, we get that for every $\ell \in [k]$, $j \in [m]$,

$$\frac{\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)}{\prod_{j \in [m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)} = e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \tag{22}$$

So, from Equations 18,19,20,21,22 we have,

$$\frac{\prod_{j \in [m]} I_{x,j}{}^{u_j} \cdot \prod_{j \in [m]} e(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j})}{\prod_{j \in [m]} e(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}) \cdot e(K_1, A_x) \cdot e(K_2, A_x) \cdot e(K_3, B_x)}$$

$$= e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot \frac{e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}}{e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}}} = e(g_q, g_q)^{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}$$

Now we consider $(y < y^*) \vee ((y, \ell, b) = (y^*, \ell^*, b^*))$. Then, we compute the following components.

$$\begin{aligned}
&\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) \\
&= \prod_{j \in [m]} e\left(g_q^{\beta \widehat{r} r_{x,j} s_x}, \prod_{\ell \in [k]} g^{c_{y,\ell,\mathsf{id}_\ell} t u_j} h^{w_{y,\ell,\mathsf{id}_\ell} \tau u_j} h^{\tau \delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell} u_j}\right) \\
&= e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot e(g_q, h_q)^{\beta \widehat{r} s_x \tau \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} (w_{y,\ell,\mathsf{id}_\ell} + \delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell})}
\end{aligned} \tag{23}$$

$$\begin{aligned}
&\prod_{j \in [m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) \\
&= \prod_{j \in [m]} e\left(h_q^{\beta \widehat{r} r_{x,j} s_x \tau}, \prod_{\ell \in [k]} g^{w_{y,\ell,\mathsf{id}_\ell} u_j} \cdot g^{\delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell} u_j} \cdot g_p^{\gamma_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell} u_j}\right) \\
&= e(h_q, g_q)^{\beta \widehat{r} s_x \tau \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} (w_{y,\ell,\mathsf{id}_\ell} + \delta_{\ell,\mathsf{id}_\ell} v_{y,\ell,\mathsf{id}_\ell})}
\end{aligned} \tag{24}$$

In this case also,

$$\frac{\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)}{\prod_{j \in [m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell \in [k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right)} = e(g_q, g_q)^{\beta \widehat{r} s_x t \langle \boldsymbol{u}, \boldsymbol{r}_x \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}}$$

So, correct decryption follows as previous.

Case 2: Otherwise. From the Tables 25 and 26, we have $R_{x,j} = E_{x,j}^{s_x}$, $\widetilde{R}_{x,j} = F_{x,j}^{s_x \tau}$, $A_x = g^{s_x t}$, $B_x = \mathrm{H}(\mathsf{gid})^{s_x t}$, and $I_{x,j} = e(g,g)^{v_j} \cdot G_{x,j}^{t s_x} \cdot e(f, Y_{x,j})^{t s_x}$, then the correctness holds if $(i^*, \ell^*) = (i, \perp) \vee (i^*, \mathsf{id}_{\ell^*}) = (i^*, 1 - \mathsf{id}_{\ell^*})$ or we can write it as $(x = x^*) \wedge ((y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*)))$

$$\prod_{j \in [m]} e\left(R_{x,j}, \prod_{\ell \in [k]} C_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j \in [m]} e\left(g^{\widehat{r} r_{x,j} s_x}, \prod_{\ell \in [k]} g^{c_{y,\ell,\mathsf{id}_\ell} t u_j} h^{w_{y,\ell,\mathsf{id}_\ell} \tau u_j}\right)$$

$$= e(g,g)^{\widehat{r}s_x t\langle \boldsymbol{u},\boldsymbol{r}_x\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} \cdot e(g,h)^{\widehat{r}s_x\tau\langle \boldsymbol{u},\boldsymbol{r}_x\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell}} \tag{25}$$

$$\prod_{j\in[m]} e\left(\widetilde{R}_{x,j}, \prod_{\ell\in[k]} \widetilde{C}_{y,\ell,\mathsf{id}_\ell}^{u_j}\right) = \prod_{j\in[m]} e\left(h^{r_{x,j}s_x\tau}, \prod_{\ell\in[k]} g^{w_{y,\ell,\mathsf{id}_\ell}u_j}\right)$$

$$= e(h,g)^{s_x\tau\langle \boldsymbol{r}_x,\boldsymbol{u}\rangle \sum_{\ell\in[k]} w_{y,\ell,\mathsf{id}_\ell}} \tag{26}$$

$$e(K_1,A_x) = e(g,g)^{s_x t\langle \boldsymbol{\alpha}_x,\boldsymbol{u}\rangle} \cdot e(g,g)^{\widehat{r}s_x t\langle \boldsymbol{r}_x,\boldsymbol{u}\rangle \sum_{\ell\in[k]} c_{y,\ell,\mathsf{id}_\ell}} \tag{27}$$

$$e(K_2,A_x) = e(f^{\langle \boldsymbol{\psi}_x,\boldsymbol{u}\rangle}\cdot \mathrm{H}(\mathsf{gid})^r, g^{s_x t}) = e(f,g)^{s_x t\langle \boldsymbol{\psi}_x,\boldsymbol{u}\rangle} \cdot e(\mathrm{H}(\mathsf{gid}),g)^{s_x tr} \tag{28}$$

$$e(K_3,B_x) = e(g^r,\mathrm{H}(\mathsf{gid})^{s_x t}) = e(g,\mathrm{H}(\mathsf{gid}))^{s_x tr} \tag{29}$$

$$\prod_{j\in[m]} {I_{x,j}}^{u_j} = e(g,g)^{\langle \boldsymbol{u},\boldsymbol{v}\rangle} \cdot e(g,g)^{s_x t\langle \boldsymbol{\alpha}_x,\boldsymbol{u}\rangle} \cdot e(f,g)^{t s_x\langle \boldsymbol{\psi}_x,\boldsymbol{u}\rangle} \tag{30}$$

From the similar computation as Case 1, the correctness of EIPL-IBIPFE holds. □

7.2 Security Analysis

This section shows the selective security of our EIPL-IBIPFE from the bilinear map. This construction is discussed in Sec. 7.

Theorem 9 *If the assumptions 1,7,4,5, and 6 hold over the bilinear group $\mathbb{BG}$, then our EIPL-IBIPFE is selectively secure as per Definition 13 to 17.*

Proof. We prove that our EIPL-IBIPFE satisfy all five security properties which we discuss in Sec. 7. We significantly modify the proof technique of [BW06, GKW19] to fit this into our scheme. Before going the main idea of the proof technique, we would like to focus that since our EIPL-IBIPFE consists of a public key special encryption algorithm thus, the adversary does not need to special encryption queries to EIPL-IBIPFE challenger. Therefore, the adversary only performs secret key queries to the challenger throughout the security game.

Lemma 11 *Our EIPL-IBIPFE satisfies normal-hiding security as per the Definition 13.*

Proof of Lemma 11. Since the distribution of normal encryption's ciphertext and special encryption's ciphertext for the index-position-bit tuple $(1,\perp,0)$ are same, thus the definition of normal-hiding security follows from scheme. □

Lemma 12 *If the assumptions 1,4,5, and 6 hold over the bilinear group $\mathbb{BG}$, then our EIPL-IBIPFE satisfies the index-hiding security as per the Definition 14.*

Proof of Lemma 12. As per the definition of index-hiding game, we show that the adversary can not distinguish between the special encryption of the index-position-value tuple $(i^*,\perp,0)$ and $(i^*+1,\perp,0)$. Note that, the adversary is not allowed to query for the secret keys corresponding to the index position $i^* = (x^*,y^*)$ and the group identity gid^* at a time.

If $y^* = \widetilde{n}$, we have $i^*+1 = (x^*+1,1)$ otherwise, $i^*+1 = (x^*,y^*+1)$. Similar to [BW06, GKW19], we consider two cases based on whether $y = \widetilde{n}$ or not. To prove this security, we consider following two claims 8 and 9.

Claim 8. For $y^* < \widetilde{n}$, the special encryption to the index-position-bit tuple $((x^*,y^*),\perp,0)$ and $((x^*,y^*+1),\perp,0)$ are indistinguishable.

Proof of claim 8. To prove the above claim, we consider $2k+1$ sequences of hybrid games as H_0 and $H_{\widetilde{\ell},\widetilde{b}}$ where $\widetilde{\ell} \in [k]$ and $\widetilde{b} \in \{0,1\}$. The hybrid H_0 corresponds to the index-hiding security game where the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*,y^*),\perp,0)$ and $H_{\widetilde{\ell},\widetilde{b}}$ is same as H_0 except the column component $C_{y^*,\ell,b}$ for $(\ell,b) \in [\widetilde{\ell}-1]\times\{0,1\}$ and for $\ell = \widetilde{\ell}, b = \widetilde{b}$,we take uniform element from $\mathbb{G}_p$.

Table 27: Computing column components of the ciphertext in Hybrid $H_{\widetilde{\ell},\widetilde{b}}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell > \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee (y = y^* \wedge \ell < \widetilde{\ell}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b \leq \widetilde{b})$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau} V^{v_{y,\ell,b}\tau}_{\ell,b}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}^{v_{y,\ell,b}}_{\ell,b}$

Here, the hybrid $H_{k,1}$ corresponds to the index-hiding game in which challenge ciphertext is a special encryption to the index-position-bit tuple $(i^*+1, \perp, 0) = ((x^*, y^*+1), \perp, 0)$ and it is also required that the hybrid H_0 and $H_{1,0}$ are indistinguishable. In the following, we show that the hybrid $H_{\widetilde{\ell},\widetilde{b}}$ and the hybrid $H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ are indistinguishable. This same proof technique are used to show all the consecutive hybrids indistinguishable. By combining all indistinguishability of hybrids, the claim 8 follows.

$H_{\widetilde{\ell},\widetilde{b}} \approx H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$: Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ that can distinguish between the hybrid $H_{\widetilde{\ell},\widetilde{b}}$ and hybrid $H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ with non-negligible advantage $\epsilon(\cdot)$. We construct a PPT reduction algorithm $\mathscr{B}$ which breaks the assumption 1 with the same advantages as follows.

Let the reduction algorithm $\mathscr{B}$ first receives the modified-1 D3DH assumption 1 challenge instances from the challenger as

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, D = g_p^{b^2}, E = g_p^{b^2 c}, F = g_p^{b^3}, G = g_p^{b^4}, H = g_p^{b^3 c}, T)$$

where T is either g_p^{abc} or a random element in the subgroup $\mathbb{G}_p$ of prime order p. Next, it receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^{k'}, 1^m, i^* = (x^*, y^*), \mathsf{gid}^*)$ from the adversary $\mathscr{A}$ where $y^* < \widetilde{n}$. Now, $\mathscr{B}$ generates the master public key by using the modified-1 D3DH instances of assumption 1 and sends it to $\mathscr{A}$. Next, the adversary makes adaptively secret keys queries for distinct indices i and the group identity gid except i^*, gid^* and sends the challenge message vector $\boldsymbol{v}$ to the challenger. In the following, we show that how does $\mathscr{B}$ generate the master public key and how to answer the queried secret keys and the challenge ciphertext from the challenge instances. Finally, $\mathscr{A}$ outputs its guess, which $\mathscr{B}$ uses to break the modified-1 D3DH assumption 1.

Since, this reduction plays over the subgroup $\mathbb{G}_p$ with its challenger, thus it can choose any required elements from the subgroup $\mathbb{G}_q$. We implicitly set the exponents as $r_{p,x^*,j} = b \cdot \widetilde{r}_{p,x^*,j}$ and $s_{p,x^*} = \widetilde{s}_{p,x^*}/b, \widehat{r}_p = b^2$ where the exponents $\widetilde{r}_{p,x^*,j}, \widetilde{s}_{p,x^*}$ are chosen uniformly random from the group $\mathbb{G}_p$. Also we set $h_p = B = g_p^b, f_p = B^{d_1}, t_p = a \cdot b, c_{p,y^*,\widetilde{\ell},\widetilde{b}} = c \cdot \widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}}$ for some uniformly chosen $d_1 \leftarrow \mathbb{Z}_N$, $\widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}} \leftarrow \mathbb{Z}_N$. With these exponents, $\mathscr{B}$ correctly simulates the master public key, secret keys and as well as the challenge group elements T that can be programmed in the challenge ciphertext components $C_{y,\widetilde{\ell},\widetilde{b}}$.

Public key simulation. The challenger $\mathscr{B}$ chooses two random generators $h_q, f_q \leftarrow \mathbb{G}_q$ such that $h_q = g_q^d, f_q = g_q^{d'}$ where the exponents $d, d' \in \mathbb{Z}_N$ are sampled randomly. Additionally, it chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, \quad & \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\} \quad & \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}] \quad & \widetilde{r}_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

Next, it samples $\beta, \widehat{r}_q \leftarrow \mathbb{Z}_N$ and computes the following public key components for all $x \in [\widehat{n}]$, $j \in [m]$ and $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$ as

$$E_{x,j} = \begin{cases} (Dg_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} \text{ if } x \neq x^*, \\ (Fg_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} \text{ otherwise .} \end{cases}, \quad F_{x,j} = \begin{cases} (Fh_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} \text{ if } x \neq x^*, \\ (Gh_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} \text{ otherwise .} \end{cases},$$

$$H_{y,\ell,b} = \begin{cases} (Cg_q)^{\widetilde{c}_{y,\ell,b}} \text{ if } (y,\ell,b) = (y^*, \widetilde{\ell}, \widetilde{b}) \\ (g_p g_q)^{\widetilde{c}_{y,\ell,b}} \text{ otherwise} \end{cases}$$

Challenger $\mathscr{B}$ samples group elements $\vartheta'_p, \vartheta_{p,i}$ from $\mathbb{G}_p$ for all $i \in [k']$ and $\vartheta'_q, \vartheta_{q,i}$ from $\mathbb{G}_q$ for all $i \in [k']$ such that $\mathrm{H}(\mathsf{gid}) = \vartheta'_p \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{p,i} \vartheta_{q,i} = \vartheta' \prod_{i \in \mathcal{V}} \vartheta_i$, $\mathrm{H}_q(\mathsf{gid}) = \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{q,i}$ where $\vartheta' = \vartheta'_p \vartheta'_q \in \mathbb{G}$, $\boldsymbol{\vartheta} = (\vartheta_i) \in \mathbb{G}^{k'}$. Finally, $\mathscr{B}$ sets the master public key as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{B}\mathbb{G}, g = g_p g_q, h = g_p^b g_q^d, f = B^{d_1} f_q, \\ \vartheta', \vartheta_q^{'\beta}, \boldsymbol{\vartheta} = (\vartheta_i), \{\vartheta_{q,i}^{\beta}\}_{i \in [k']}, \mathrm{H}, \mathrm{H}_q, E_q = g_q^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = g_q^{\beta \widehat{r}_q \widetilde{r}_{x,j}}, F_{q,x,j} = h_q^{\beta \widehat{r}_q \widetilde{r}_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}}, \\ Y_{x,j} = g^{\psi_{x,j}}, Y_{q,x,j} = g_q^{\beta \psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]}, \\ \{H_{y,\ell,b}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right),$$

All the public key components can be computed using the challenge instances of modified-1 D3DH assumption 1.

Secret Key simulation. To answer these queries, challenger returns the secret key $\mathsf{sk}_{\boldsymbol{u}}$ corresponding to the tuple $(i = (x,y), \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ as follows: Note that the adversary is not allowed to secret key queries corresponding to the tuple $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ to the key generation oracle.

If $\mathsf{gid} = \mathsf{gid}^*$, then adversary can not query for the secret key corresponding to the index i^*. To generate the secret keys the challenger computes $\mathrm{H}(\mathsf{gid}^*) = (\vartheta'_p \vartheta'_q \prod_{i \in \mathcal{V}^*} \vartheta_{p,i} \vartheta_{q,i}) = g_p^{d_1^*} g_q^{d_2^*}$ where $\mathcal{V}^*$ associated with the non-zero indices associated with the group identity gid^* and $d_1^*, d_2^* \leftarrow \mathbb{Z}_N$. The challenger $\mathscr{B}$ chooses a random value $\widetilde{r} \leftarrow \mathbb{Z}_N$ and set $r = \widehat{r} \cdot \widetilde{r}$. Then it simulates the secret keys as follows:

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} (Dg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x \neq x^*, y \neq y^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} (Fg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x = x^*, (y \neq y^* \vee \mathsf{id}_\ell \neq \widetilde{b}), \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} (Dg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \widetilde{\ell}} c_{y,\ell,\mathsf{id}_\ell}} (Eg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\widetilde{\ell},\mathsf{id}_{\widetilde{\ell}}}} & \text{if } x \neq x^* \wedge (y, \mathsf{id}_{\widetilde{\ell}}) = (y^*, \widetilde{b}), \end{cases}$$

$$K_2 = (B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}} \text{ for all } x \in [\widehat{n}], \quad K_3 = (Dg_q^{\widehat{r}_q})^{\widetilde{r}}$$

For $\mathsf{gid} = \mathsf{gid}^*$, adversary can query for the secret key corresponding to the index i^*. In the following, $\mathscr{B}$ generates the secret keys for this case.

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} (F^{\sum_{\ell \neq \widetilde{\ell}} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} H^{\widetilde{c}_{y,\widetilde{\ell},\mathsf{id}_{\widetilde{\ell}}}})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle} & \text{if } x = x^*, y = y^*, \mathsf{id}_{\widetilde{\ell}} = \widetilde{b} \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} F^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } x = x^*, y = y^*, \mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b} \end{cases}$$

Without loss of generality, we assume that the adversary makes the maximum number of Q queries with the challenge group identity gid^* and challenge index i^*. Now, the simulator chooses an integer $k_1' \leftarrow [k']$, sets an integer $s = 10Q$, a random k'-length vector $\boldsymbol{z} = (z_i) \leftarrow \mathbb{Z}_s^{k'}$ and a value $z' \leftarrow \mathbb{Z}_s$. Additionally, the simulator also chooses a random value $w' \leftarrow \mathbb{Z}_N$ and an uniformly random k'-length vector $\boldsymbol{w} = (w_i) \leftarrow \mathbb{Z}_N^{k'}$. All these values are kept secret to the simulator.

Let us consider $\mathcal{V}^* \subseteq \{1, 2, \ldots, k'\}$ be the set of all i for which the challenge identity $\mathrm{gid}_i^* = 1$. Let $\mathcal{V}^* = \{i_1, i_2, \ldots, i_\kappa\}$. Now, we choose the z_i values from $\boldsymbol{z}$ which correspond to the collection of indices $\mathcal{V}^*$. Then set $\sum_{i \in \mathcal{V}^*} z_i = k_1' s - z'$ for uniformly chosen $k_1' \in [k']$. Now, we define the function K(gid) as

$$\mathrm{K}(\mathrm{gid}) = \begin{cases} 0, & \text{if } z' + \sum_{i \in \mathcal{V}} z_i \equiv 0 \mod s \\ 1, & \text{elsewhere} \end{cases}$$

So, from the above definition of the function K, we can say that $\mathrm{K}(\mathrm{gid}^*) = 0$ and for all $\mathrm{gid}(\neq \mathrm{gid}^*)$ it becomes non-zero. Additionally, we set two functions as $\mathrm{F}(\mathrm{gid}) = N - sk_1' + z' + \sum_{i \in \mathcal{V}} z_i$ and $\mathrm{J}(\mathrm{gid}) = w' + \sum_{i \in \mathcal{V}} w_i$. The simulator assigns the public parameters $\vartheta' = f^{N - k_1' s + z'} \cdot g^{w'} = g_p^{d_p'} g_q^{d_q'}$ and $\vartheta_i = f^{z_i} g^{w_i} = g_p^{d_{p,i}} g_q^{d_{q,i}}$. Now $\mathcal{B}$ answers secret key components K_2, K_3 as follows:

$$\begin{aligned}
K_2 &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathrm{gid})}{\mathrm{F}(\mathrm{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}} \\
&= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathrm{gid})}{\mathrm{F}(\mathrm{gid})}} \mathrm{H}(\mathrm{gid})^r \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathrm{gid})} g^{\mathrm{J}(\mathrm{gid})} \right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathrm{gid})}} \left(f^{\mathrm{F}(\mathrm{gid})} g^{\mathrm{J}(\mathrm{gid})} \right)^r \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathrm{gid})} g^{\mathrm{J}(\mathrm{gid})} \right)^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathrm{gid})}} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathrm{gid})} g^{\mathrm{J}(\mathrm{gid})} \right)^{r'} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathrm{gid})^{r'} \\
K_3 &= (D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathrm{gid})}} = g^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathrm{gid})}} = g^{r'}
\end{aligned}$$

We implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathrm{gid})}$. So from the construction of K function, we get $\mathrm{K}(\mathrm{gid}) \neq 0$ for any key query corresponding to the group identity gid. This implies that the function $\mathrm{F}(\mathrm{gid}) \neq 0 \mod N$ for any identity (since we assume $N > sk_1'$ for any reasonable value of N, s and k_1'). To prove this, we need the following Lemma 16.

Challenge ciphertext simulation. $\mathcal{B}$ makes the simulation of the challenge ciphertext as follows:

$$\begin{aligned}
\forall\, j \in [m], \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N,\ \tau \in \mathbb{Z}_N,\ t_q \leftarrow \mathbb{Z}_N, \\
\forall\, x \in [\widehat{n}], \quad e_x, f_x, d_x \leftarrow \mathbb{Z}_N, \widetilde{\kappa}_x, \widetilde{s}_x \leftarrow \mathbb{Z}_N, \\
\forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad \widetilde{w}_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N,
\end{aligned}$$

For the challenge group identity gid^*, $\mathcal{B}$ computes $\mathrm{H}(\mathrm{gid}^*) = (\vartheta_p' \vartheta_q' \prod_{i \in \mathcal{V}^*} \vartheta_{p,i} \vartheta_{q,i}) = g_p^{d_1^*} g_q^{d_2^*}$ for some $d_1^*, d_2^* \in \mathbb{Z}_N$ and $\mathrm{H}_q(\mathrm{gid}^*)^\beta = \vartheta_q^{'\beta} \prod_{i \in \mathcal{V}^*} \vartheta_{q,i}^\beta$. Now, for all $x \in [\widehat{n}]$, $j \in [m]$, challenger computes the row and columns components as follows:

Table 28: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$.

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{\widetilde{s}_x}$	$F_{q,x,j}^{\widetilde{s}_x \tau}$	$E_q^{\widetilde{s}_x t_q}$	$\mathrm{H}_q(\mathsf{gid}^*)^{\beta \widetilde{s}_x t_q}$	$e(g_q, g_q)^{v_j} \cdot G_{q,x,j}^{t_q \widetilde{s}_x} \cdot e(f_q, Y_{q,x,j})^{t_q \widetilde{s}_x}$
$x = x^*$	$D^{\widetilde{r}_{x,j} \widetilde{s}_x} g_q^{\widehat{r}_q \widetilde{r}_{x,j} \widetilde{s}_x}$	$F^{\widetilde{r}_{x,j} \widetilde{s}_x \tau} g_q^{\widehat{r}_q \widetilde{r}_{x,j} \widetilde{s}_x \tau}$	$(A g_q^{t_q})^{\widetilde{s}_x}$	$(A^{d_1^*} g_q^{d_2^* t_q})^{\widetilde{s}_x}$	$e(g,g)^{v_j} \cdot e(g, A g_q^{t_q})^{\alpha_{x,j}, \widetilde{s}_x} \cdot e(A,B)^{d_1 \psi_{x,j} \widetilde{s}_x} \cdot e(f_q, g_q)^{\psi_{x,j} t_q \widetilde{s}_x}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$(Bh_q)^{\widetilde{s}_x \tau v_j}$	g^{e_x}	$(g_p^{d_1^*} g_q^{d_2^*})^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 29: Computing the columns components for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell > \widetilde{\ell}) \vee (y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\widetilde{w}_{y,\ell,b} \tau}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} \cdot g^{\widetilde{w}_{y,\ell,b}}$
$y = y^* \wedge \ell = \widetilde{\ell} \wedge b = \widetilde{b}$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot T^{\widetilde{c}_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$
$(y < y^*) \vee (y = y^* \wedge \ell < \widetilde{\ell}) \vee (y = y^* \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot g_p^{v_{y,\ell,b}}$	$g^{w_{y,\ell,b}}$

After generating all the ciphertext components, challenger sends these to the adversary $\mathscr{A}$, then $\mathscr{A}$ guess a bit $\mathfrak{b}'$ and sends it to $\mathscr{B}$. It simply forwards it as the guess to the modified-1 D3DH challenger of assumption 1.

Analysis of simulation. If $T = g_p^{abc}$, then $\mathscr{B}$ simulates the view of the hybrid is similar as $H_{\widetilde{\ell},\widetilde{b}}$ otherwise if T is random group elements from $\mathbb{G}_p$ the view of the hybrid is similar as $H_{\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \bmod 2}$. Thus, if $\mathscr{A}$ wins with the advantages $\epsilon(\cdot)$ then $\mathscr{B}$ breaks the modified-1 D3DH assumption 1 with the same advantages.

Claim 9. If $y^* = \widetilde{n}$, then the special encryption to the index-position-bit tuple $(x^*, y^*, \perp, 0)$ and $((x^*+1, 1), \perp, 0)$ are indistinguishable.

Proof of claim 9. To prove the above claim, we consider a sequence of hybrids games. In the following, we discuss about these hybrids.

Hybrid 1. This hybrid corresponds to the index-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$ for $y^* = \widetilde{n}$.

Hybrid 2. The hybrid is same as hybrid 1 except that the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*+1), \perp, 0)$ for $y^* = \widetilde{n}$. Note that, the special-encryption algorithm does not generally encrypt to position $(x^*, y^*+1 = \widetilde{n}+1)$, however the algorithm can be naturally extended to encrypt to such position.

Hybrid 3. Hybrid 3 is same as the previous hybrid 2 except that the row component $I_{x^*,j}$ as mentioned in the following Table 30.

Table 30: Computing row components for the ciphertext $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t}$	$\mathrm{H}_q(\mathsf{gid}^*)^{\beta s_x t}$	$e(g_q, g_q)^{v_j} \cdot G_{q,x,j}^{t s_x} \cdot Y_{q,x,j}^{t s_x}$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$g^{s_x t}$	$\mathrm{H}(\mathsf{gid}^*)^{s_x t}$	$e(g,g)^{v_j} \cdot G_{x,j}^{t s_x} \cdot Y_{x,j}^{t s_x} \cdot L$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau v_j}$	g^{e_x}	$\mathrm{H}(\mathsf{gid}^*)^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

where $L = e(g_p, g)^z$ and z is randomly chosen from $\mathbb{Z}_p$.

Hybrid 4. Hybrid 4 is identical to hybrid 3 except that the row component of the challenge ciphertext as the Table 30. Here we consider $L = e(g,g)^z$ with z is a random exponent from $\mathbb{Z}_N$.

Hybrid 5. Hybrid 5 is the same as hybrid 4 except that row component in the challenge ciphertext as in Table 31 mentioned below.

Table 31: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t}$	$\mathrm{H}_q(\mathsf{gid}^*)^{\beta s_x t}$	$e(g_q,g_q)^{v_j} \cdot G_{q,x,j}^{t s_x} \cdot Y^{t s_x}_{q,x,j}$
$x \le x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau v_j}$	g^{e_x}	$\mathrm{H}(\mathsf{gid}^*)^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Hybrid 6. The hybrid 6 is similar to the hybrid 5 except that the column components to the index-position-bit tuple $((x^*, y^* = 1), \ell^* = \perp, b^* = 0)$ of the challenge ciphertext.

Hybrid 7. The hybrid 7 corresponds to the index-hiding security game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $((x^*+1, 1), \perp, 0)$ for $y^* = \widetilde{n}$.

In the following, we prove that the adversary's advantage for all the consecutive hybrids is negligible in the security parameter which completes the proof of the claim 9.

Hybrid 1 $\approx$ Hybrid 2: The indistinguishable proof of the hybrid 1 and hybrid 2 is identical to the claim 8.

Hybrid 2 $\approx$ Hybrid 3: Suppose on the contrary, there exists a PPT adversary $\mathscr{A}$ that distinguishes between the above two hybrids with the non-negligible advantages $\epsilon(\lambda)$. We construct a PPT reduction algorithm that breaks the modified-1 D3DH assumption 1 with same non-negligible advantages as follows:

The reduction algorithm $\mathscr{B}$ first receives the DBDH challenges from its challenger as given below.

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, T = e(g_p,g)^z)$$

where z is either abc or a random element from $\mathbb{Z}_N$. Next, in the setup phase, adversary receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^m, i^* = (x^*, y^*), \mathsf{gid}^*)$ from the adversary $\mathscr{A}$ satisfying the condition $y^* = \widetilde{n}$. Since, the reduction game plays with its challenger in the subgroup $\mathbb{G}_p$, thus it can choose any elements from $\mathbb{G}_q$ by itself. Now, $\mathscr{B}$ generates the master public key using the given instances and sends it to the adversary $\mathscr{A}$. Then, the adversary can not make secret keys query corresponding the index i^* and gid^* at a time. In the following, we show how does $\mathscr{B}$ simulate the master public key, secret keys and challenge ciphertext from the given instances. Finally, $\mathscr{A}$ outputs its guess, which is used to break the DBDH assumption. For $x = x^*$, our approach is to implicitly set the exponents $r_{p,x^*,j} = b\mathfrak{r}_j, \alpha_{p,x^*,j} = abk\widehat{r}\mathfrak{r}_j, t_p = c$ where $\mathfrak{r}_j \leftarrow \mathbb{Z}_N$ for all $j \in [m]$. Additionally, we implicitly set $c_{y,\ell,b} = \widetilde{c}_{p,y,\ell,b} - a$ for all $(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}$. According to the exponents as given above, the challenger simulates the master public key, the secret keys and the challenge ciphertext components.

Public key simulation. To generate the master public key, it chooses the random exponents as follows:

$$\begin{aligned}
\forall \ell \in [k], b \in \{0,1\}, \quad & \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p, \\
\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad & \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N, \widehat{r} \leftarrow \mathbb{Z}_N, \\
\forall j \in [m], x \in [\widehat{n}], \quad & \widetilde{r}_{x,j}, \widetilde{\alpha}_{x,j}, \widetilde{\psi}_{x,j} \leftarrow \mathbb{Z}_N,
\end{aligned}$$

Additionally, it samples random group elements $\vartheta'_p, \vartheta_{p,i} \in \mathbb{G}_p$ for all $i \in [k']$, $\vartheta'_q, \vartheta_{q,i} \in \mathbb{G}_q$ for all $i \in [k']$ such that $\vartheta' = \vartheta'_p \vartheta'_q, \boldsymbol{\vartheta} = (\vartheta_{p,i}\vartheta_{q,i})_{i\in[k']} \in \mathbb{G}^{k'}$, and also choose random integers $d, d_1, \beta \leftarrow \mathbb{Z}_N$ for all $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$. Then $\mathscr{B}$ computes

$$E_{x,j} = \begin{cases} (g_p g_q)^{\widehat{r}\widetilde{r}_{x,j}} & \text{if } x \neq x^* \\ (B^{\mathfrak{r}_j} g_q^{\widetilde{r}_{x,j}})^{\widehat{r}} & \text{otherwise} \end{cases}, F_{x,j} = \begin{cases} h^{\widehat{r}\widetilde{r}_{x,j}} & \text{if } x \neq x^* \\ (B^{\mathfrak{r}_j d} g_q^{d\widetilde{r}_{x,j}})^{\widehat{r}} & \text{otherwise} \end{cases}$$

$$G_{x,j} = \begin{cases} e(g_p g_q, g_p g_q)^{\widetilde{\alpha}_{x,j}} & \text{if } x \neq x^* \\ e(A,B)^{k\widehat{r}\mathfrak{r}_j} e(g_q, g_q)^{\widetilde{\alpha}_{x,j}} & \text{otherwise} \end{cases}$$

Sets the master public key

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = g^d, f = g^{d_1}, \\ \vartheta_q^{'\beta}, \vartheta', \boldsymbol{\vartheta} = (\vartheta_i), \{\vartheta_{q,i}^{\beta}\}_{i\in[k']}, \mathrm{H}, \mathrm{H}_q, E_q = g_q^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j}, E_{q,x,j} = g_q^{\beta\widehat{r}\widetilde{r}_{x,j}}, \\ F_{q,x,j} = h^{\beta\widehat{r}\widetilde{r}_{x,j}}, G_{q,x,j} = e(g_q,g_q)^{\beta\widetilde{\alpha}_{x,j}}, \\ Y_{x,j} = g^{\widetilde{\psi}_{x,j}}, Y_{q,x,j} = g_q^{\beta\widetilde{\psi}_{x,j}} \end{array} \right\}_{x\in[\widehat{n}], j\in[m]}, \\ \{H_{y,\ell,b} = A^{-1} g^{\widetilde{c}_{y,\ell,b}}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b)\in[k]\times\{0,1\}} \end{array} \right),$$

Secret key simulation. The challenger $\mathscr{B}$ generates the secret key $\mathsf{sk}_{\boldsymbol{u}}$ corresponding to the adversary's query tuple $(i = (x, y), \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ as below.

First consider $\mathsf{gid} = \mathsf{gid}^*$, then the adversary can not query for the secret key associated with the index position i^*.

$$K_1 = \begin{cases} g^{\langle\widetilde{\boldsymbol{\alpha}}_x, \boldsymbol{u}\rangle + \widehat{r}\langle\widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} A^{-\widehat{r}\langle\widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle} & \text{if } x \neq x^* \\ g_q^{\langle\widetilde{\boldsymbol{\alpha}}_x, \boldsymbol{u}\rangle + \widehat{r}\langle\widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} B^{\widehat{r}\langle\mathfrak{r}, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{otherwise} \end{cases}$$

$$K_2 = g^{d_1\langle\widetilde{\boldsymbol{\psi}}_x, \boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid})^r \ \forall x \in [\widehat{n}], \ \ K_3 = g^r$$

where $r = \widehat{r} \cdot \widetilde{r}$ and $\widetilde{r}$ is randomly chosen from $\mathbb{Z}_N$. Note that, the adversary $\mathscr{A}$ is not allowed to query for the secret key corresponding to the pair $(i^* = (x^*, y^*), \mathsf{gid}^*)$.

If $\mathsf{gid} \neq \mathsf{gid}^*$, then the adversary can query for the index position i^*. Then the challenger generates the corresponding secret keys as below

$$K_1 = g_q^{\langle\widetilde{\boldsymbol{\alpha}}_x, \boldsymbol{u}\rangle + \widehat{r}\langle\widetilde{\boldsymbol{r}}_x, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{\widetilde{n},\ell,\mathsf{id}_\ell}} B^{\widehat{r}\langle\mathfrak{r}, \boldsymbol{u}\rangle \sum_{\ell\in[k]} \widetilde{c}_{\widetilde{n},\ell,\mathsf{id}_\ell}} \text{ if } x = x^*$$

As the previous case, we assume that the adversary makes the maximum number of Q queries, the challenge group identity gid^* and challenge index i^*. Now, the simulator chooses an integer $k'_1 \leftarrow [k']$, sets an integer $s = 10Q$, a random k'-length vector $\boldsymbol{z} = (z_i) \leftarrow \mathbb{Z}_s^{k'}$ and a value $z' \leftarrow \mathbb{Z}_s$. Additionally, the simulator chooses a random value $w' \leftarrow \mathbb{Z}_N$ and an uniformly random k'-length vector $\boldsymbol{w} = (w_i) \leftarrow \mathbb{Z}_N^{k'}$. All these values are kept secret to the $\mathscr{B}$.

Let us consider $\mathcal{V}^* \subseteq \{1, 2, \ldots, k'\}$ be the set of all i for which the challenge identity $\mathsf{gid}_i^* = 1$. Let $\mathcal{V}^* = \{i_1, i_2, \ldots, i_\kappa\}$. Now, we choose the z_i values from $\boldsymbol{z}$ which correspond to the collection of indices $\mathcal{V}^*$. Sets $\sum_{i\in\mathcal{V}^*} z_i = k'_1 s - z'$ for uniformly chosen $k'_1 \in [k']$. Now, we define the function K(gid) as

$$\mathrm{K}(\mathsf{gid}) = \begin{cases} 0, & \text{if } z' + \sum_{i\in\mathcal{V}} z_i \equiv 0 \mod s \\ 1, & \text{elsewhere} \end{cases}$$

So, from the above definition of the function K, we can say $\mathrm{K}(\mathsf{gid}^*) = 0$ and for all $\mathsf{gid}(\neq \mathsf{gid}^*)$ it becomes non-zero. Additionally, we set two functions as $\mathrm{F}(\mathsf{gid}) = N - sk_1' + z' + \sum_{i\in\mathcal{V}} z_i$ and $\mathrm{J}(\mathsf{gid}) = w' + \sum_{i\in\mathcal{V}} w_i$. The simulator assigns the public parameters $\vartheta' = f^{N-k_1's+z'} \cdot g^{w'} = g_p^{d_p'} g_q^{d_q'}$ and $\vartheta_i = f^{z_i} g^{w_i} = g_p^{d_{p,i}} g_q^{d_{q,i}}$. Now $\mathscr{B}$ answers remaining secret key components as

$$
\begin{aligned}
K_2 &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \mathrm{H}(\mathsf{gid})^r \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r'} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid})^{r'} \\
K_3 &= g^r \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r'}
\end{aligned}
$$

We implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}$. So from the construction of K function, we get $\mathrm{K}(\mathsf{gid}) \neq 0$ for any key query corresponding to the group identity gid. This implies that the function $\mathrm{F}(\mathsf{gid}) \neq 0 \mod N$ for any identity (as we assume $N > sk_1'$ for any reasonable value of N, s and k_1') To prove this, we need the following Lemma 16.

Challenge ciphertext simulation. The challenger $\mathscr{B}$ can compute all the column components corresponding to $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$ on its own, since $\mathbb{G}_p$ subgroup components are random in $C_{y,\ell,b}, \widetilde{C}_{y,\ell,b}$ terms and for computing remaining terms over the subgroup $\mathbb{G}_q$, the required exponents are already known to the challenger $\mathscr{B}$. For $x < x^*$, all the row components $R_{x,j}, \widetilde{R}_{x,j}, A_x, B_x, I_{x,j}$ are chosen randomly, but for $x > x^*$, all the row components are formed over the subgroup $\mathbb{G}_q$ which it knows. For the challenge group identity gid^*, the challenger computes $\mathrm{H}(\mathsf{gid}^*) = (\vartheta_p' \vartheta_q') \prod_{j\in\mathcal{V}^*} (\vartheta_{p,j} \vartheta_{q,j}) = g_p^{d_1^*} g_q^{d_2^*}$ for some $d_1^*, d_2^* \in \mathbb{Z}_N$. Then the challenger $\mathscr{B}$ simulates the challenge ciphertext for $x = x^*$ as follows:

Table 32: Row component generation for $x = x^*$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x = x^*$	$(B^{\mathfrak{r}_j} g_q^{\widetilde{r}_{x,j}})^{\widehat{r}\widetilde{s}_x}$	$(B^{\mathfrak{r}_j} g_q^{\widetilde{r}_{x,j}})^{d\widehat{r}\widetilde{s}_x\tau}$	$(C g_q^{t_q})^{\widetilde{s}_x}$	$(C^{d_1^*} g_q^{t_q d_2^*})^{\widetilde{s}_x}$	$e(g,g)^{\nu_j} \cdot e(g_q, g_q)^{\widetilde{\alpha}_{x,j}\widetilde{s}_x t_q}$ $\cdot T^{\widehat{r}k\widetilde{s}_x \mathfrak{r}_j} \cdot e(f, C g_q^{t_q})^{\widetilde{\psi}_{x,j}\widetilde{s}_x}$

where the exponents $\widetilde{s}_{x^*}, \tau, t_q$ are randomly sampled from $\mathbb{Z}_N$. Finally, $\mathscr{B}$ gets the guess bit $\mathfrak{b}'$ from $\mathscr{A}$ and it simply forwarded it to the DBDH challenger.

Analysis of simulation. If $T = e(g_p, g)^{abc}$, then $\mathscr{B}$ simulates the view of hybrid 2 else if $T = e(g_p, g)^z$ for any random z from $\mathbb{Z}_N$, adversary's view same as hybrid 3. Therefore, if $\mathscr{A}$ wins the game with advantages $\epsilon(\cdot)$, then $\mathscr{B}$ breaks the DBDH assumption with the same advantages.

Hybrid 3 ≈ Hybrid 4: To show the indistinguishability of two the hybrids 3 and 4, we use a similar proof technique of [BW06, GKW19]. Here, we discuss the underlying approaches. Let us consider that $\mathscr{B}$ receives the Bilinear Subgroup Decisional (BSD) assumption 5 challenge instances from the challenger consisting the bilinear group $\mathbb{BG}, e(T, g)$ where T is either a random element from the subgroup $\mathbb{G}_p$ or a uniform element from the group $\mathbb{G}$. Then $\mathscr{B}$

computes all the components for the master public key mpk honestly and forwarded it to the adversary. After seeing mpk, adversary can query for the secret key to the key generation oracle. Finally, $\mathscr{B}$ computes all the challenge ciphertext components honestly except the value $I_{x^*,j} = e(g,g)^{v_j^{(\mathfrak{b})}} \cdot G_{x^*,j}^{s_{x^*}t} \cdot e(g,T) \cdot e(f,g)^{\psi_{x^*,j} s_{x^*} t}$ where T is taken from BSD challenge of assumption 1. If $T \leftarrow \mathbb{G}_p$, then simulator's view is same as hybrid 3 otherwise, if $T \leftarrow \mathbb{G}$ then $\mathscr{B}$ perfectly simulates as hybrid 4. Therefore, if $\mathscr{A}$ wins with the advantage $\epsilon(\cdot)$, then $\mathscr{B}$ breaks the assumption 5 with same advantage $\epsilon(\cdot)$.

Hybrid 4 ≈ Hybrid 5: Suppose on the contrary, there exists PPT adversary $\mathscr{A}$ that distinguish between the hybrid 4 and hybrid 5 with the non-negligible advantage $\epsilon(\cdot)$. Then, we construct a PPT reduction algorithm which breaks the R3DH assumption 6 with same non-negligible advantages.

Let the reduction algorithm $\mathscr{B}$ first receives the challenges of R3DH assumption 6 from the challenger as

$$(\mathbb{BG}, \mathfrak{g}_p \in \mathbb{G}_p, \mathfrak{g}_q \in \mathbb{G}_q, A = \mathfrak{g}_q^a, B = \mathfrak{g}_p^{\widetilde{a}} \cdot \mathfrak{g}_q^{a^2}, C = \mathfrak{g}_p^{\widetilde{c}} \cdot \mathfrak{g}_q^{c}, D = \mathfrak{g}_p^{\widetilde{a}\widetilde{c}}, T)$$

where T is either $\mathfrak{g}_q^{a^2 c}$ or a random element from the subgroup $\mathbb{G}_q$. Next, the challenger $\mathscr{B}$ receives the challenge tuple $(1^n, 1^k, 1^{k'}, 1^m, i^* = (x^*, y^*), \mathsf{gid}^*)$ from $\mathscr{A}$. Then, $\mathscr{B}$ generates the public keys and sends it to the adversary. After getting the public parameters, $\mathscr{A}$ can adaptively query to the key generation oracle corresponding for the tuple $(i = (x,y), \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$. Next the adversary uniformly chooses a challenge message vector pair $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ and sends it to $\mathscr{B}$. To answer the challenge ciphertext, $\mathscr{B}$ randomly chooses a bit $\mathfrak{b} \in \{0,1\}$ and generates the challenge ciphertext $\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})}$. In the following, we describe how does the challenger simulate the master public key, the secret key and the challenge ciphertext using the assumption 6 instances. Finally, the adversary $\mathscr{A}$ outputs a guess bit which breaks the assumption 6. As, the reduction plays the game with the challenger in the subgroup $\mathbb{G}_q$, so it chooses all the components from the subgroup $\mathbb{G}_p$ by itself. Although, in the challenge instances of B, C some parts belong to the subgroup $\mathbb{G}_p$ but their exponents depends on $\widetilde{a}$ and $\widetilde{c}$ terms. In the following, we implicitly set the exponents as

$$g_p = \mathfrak{g}_p, g_q = A, \qquad r_{q,x^*,j} = \widetilde{r}_{q,x^*,j}/a, \qquad r_{p,x^*,j} = \widetilde{r}_{p,x^*,j},$$
$$s_{q,x^*} = c, \qquad s_{p,x^*} = \widetilde{c}, \qquad t_q = a, t_p = \widetilde{a},$$
$$\text{for all } x \in [\widehat{n}] - \{x^*\}, s_x = \widetilde{s}_x / a$$

where $\widetilde{r}_{p,x^*,j}, r_{q,x^*,j} \leftarrow \mathbb{Z}_N$ and for all $x \in [\widehat{n}] - \{x^*\}, \widetilde{s}_x \leftarrow \mathbb{Z}_N$. Additionally, the reduction algorithm samples the exponents uniformly random from $\mathbb{Z}_N$. Note that, the reduction algorithm does not know the factorization so at any point, we do not sample these exponents from $\mathbb{G}_p$ and $\mathbb{G}_q$ separately, but instead of sample any exponents directly from $\mathbb{Z}_N$ and make sure that the distributions are not affected.

Public key simulation. To generate the public key, $\mathscr{B}$ chooses the random exponents as follows:

$$\forall \ell \in [k], b \in \{0,1\}, \quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p$$
$$\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad c_{y,\ell,b} \leftarrow \mathbb{Z}_N$$
$$\forall j \in [m], x \in [\widehat{n}], \quad \widetilde{r}_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N$$

Also, it chooses the exponents $d, \beta, \widehat{r} \leftarrow \mathbb{Z}_N$ to compute all the remaining components of the master public key. For all $x \in [\widehat{n}], j \in [m]$ and $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$, $\mathscr{B}$ computes the following components.

$$E_{q,x,j} = \begin{cases} A^{\beta \widehat{r} \widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ \mathfrak{g}_q^{\beta \widehat{r} \widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases}, \quad E_{x,j} = \begin{cases} (\mathfrak{g}_p A)^{\widehat{r} \widetilde{r}_{x,j}} & \text{if } x \neq x^*. \\ (\mathfrak{g}_p \mathfrak{g}_q)^{\widehat{r} \widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases},$$

$$\begin{aligned} F_{q,x,j} &= E_{q,x,j}^d, \quad G_{q,x,j} = e(A,A)^{\beta \alpha_{x,j}}, \qquad Y_{q,x,j} = A^{\beta \psi_{x,j}}, \\ F_{x,j} &= E_{x,j}^d, \qquad G_{x,j} = e(\mathfrak{g}_p A, \mathfrak{g}_p A)^{\alpha_{x,j}}, \quad Y_{x,j} = (\mathfrak{g}_p A)^{\psi_{x,j}}, \end{aligned}$$

Also, we choose random group elements $\vartheta'_p, \vartheta_{p,i} \in \mathbb{G}_p$ for all $i \in [k']$ and $\vartheta'_q, \vartheta_{q,i} \in \mathbb{G}_q$ for all $i \in [k']$ such that $\mathrm{H}(\mathsf{gid}) = (\vartheta'_p \vartheta'_q) \prod_{i \in \mathcal{V}} (\vartheta_{p,i} \vartheta_{q,i}) = \vartheta' \prod_{i \in \mathcal{V}} \vartheta_i$ and $\mathrm{H}_q(\mathsf{gid}) = \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{q,i}$ corresponding to any group identity gid. Now, the challenger $\mathscr{B}$ sets master public key as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = \mathfrak{g}_p A, h = g^d, f = g^{d_1}, \\ \vartheta_q^{'\beta}, \vartheta', \boldsymbol{\vartheta} = (\vartheta_i), \{\vartheta_{q,i}^{\beta}\}_{i \in [k']}, \mathrm{H}, \mathrm{H}_q, E_q = A^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j}, Y_{x,j}, \\ E_{q,x,j}, F_{q,x,j}, G_{q,x,j}, Y_{q,x,j} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]}, \\ \{H_{y,\ell,b} = (\mathfrak{g}_p A)^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

Secret key simulation. First, $\mathscr{B}$ computes $\mathrm{H}(\mathsf{gid}) = \vartheta' \prod_{i \in \mathcal{V}} \vartheta_i = g^{d'} = (\mathfrak{g}_p A)^{d'}$ for some $d' \leftarrow \mathbb{Z}_N$. In the query phase, the adversary $\mathscr{A}$ is not allowed to query for the challenge index and group identity $i^* = (x^*, y^*), \mathsf{gid}^*$ together. If $\mathsf{gid} = \mathsf{gid}^*$, $\mathscr{B}$ answers the secret key $\mathsf{sk}_{\boldsymbol{u}}$ corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ as given below. Note that, the adversary is not allowed to secret key query for the index position i^*.

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (\mathfrak{g}_p A)^{\widehat{r} \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} & \text{if } x \neq x^*, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (\mathfrak{g}_p \mathfrak{g}_q)^{\widehat{r} \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} & \text{otherwise.} \end{cases},$$

$$K_2 = (\mathfrak{g}_p A)^{d_1 \langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathsf{gid})^r, \quad K_3 = g^r.$$

If $\mathsf{gid} \neq \mathsf{gid}^*$, then the secret keys corresponding to the index i^* looks as

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (\mathfrak{g}_p \mathfrak{g}_q)^{\widehat{r} \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} c_{y,\ell,\mathsf{id}_\ell}} \text{ if } x = x^*, y = y^* = \widetilde{n}$$

To simulate the keys components K_2 and K_3, challenger similarly constructs the function F, J and K functions as mentioned in the hybrid 2 to hybrid 3 indistinguishability proof. Also, $\mathscr{B}$ sets similar public key parameters ϑ' and ϑ_i's. It answers the remaining secret key components as follows:

$$\begin{aligned} K_2 &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \mathrm{H}(\mathsf{gid})^r \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^r \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r'} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathsf{gid})^{r'} \\ K_3 &= g^r \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r'} \end{aligned}$$

As in the previous argument, we implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F(gid)}}$. Therefore, from the construction of K function, we get K(gid) $\neq 0$ for any key query corresponding to the group identity gid. This implies the function F(gid) $\neq 0 \mod N$ for any identity (since we assume $N > sk_1'$ for any reasonable value of N, s and k_1'). To prove this, we need the following Lemma 16.

Challenge ciphertext simulation. To generate the challenge ciphertext, $\mathscr{B}$ chooses the random exponents as follows

$$\begin{aligned} \forall\, j \in [m], \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N,\ \tau, \widetilde{t}_p \in \mathbb{Z}_N, \\ \forall\, x \in [\widehat{n}],\quad e_x, f_x, d_x \leftarrow \mathbb{Z}_N, \widetilde{s}_x \leftarrow \mathbb{Z}_N, \\ \forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\},\quad \widetilde{w}_{y,\ell,b}, \widetilde{v}_{y,\ell,b} \leftarrow \mathbb{Z}_N. \end{aligned}$$

For the challenge identity gid*, $\mathscr{B}$ computes $\mathrm{H(gid^*)} = (\vartheta') \prod_{i \in \mathscr{V}^*} \vartheta_i = g_p^{d_1^*} g_q^{d_2^*} = \mathfrak{g}_p^{d_1^*} A^{d_2^*}$ and $\mathrm{H}_q(\mathrm{gid}^*)^\beta = \vartheta_q^{'\beta} \prod_{i \in \mathscr{V}^*} \vartheta_{q,i}^{\beta} = g_q^{\beta d_2^*} = A^{\beta d_2^*}$ for some $d_1^*, d_2^* \in \mathbb{Z}_N$. Now, for all $x \in [\widehat{n}]$, challenger computes the row and column components as follows.

Table 33: Computing row components of the ciphertext for $x \in [\widehat{n}], j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$\mathfrak{g}_q^{\beta \widehat{r} \widetilde{r}_{x,j} \widetilde{s}_x}$	$\mathfrak{g}_q^{\beta \widehat{r} \tau d \widetilde{r}_{x,j} \widetilde{s}_x}$	$A^{\beta \widetilde{s}_x}$	$A^{d_2^* \beta \widetilde{s}_x}$	$e(g_q, g_q)^{\nu_j} e(A,A)^{\beta \alpha_{x,j} \widetilde{s}_x} e(A,A)^{d_1 \beta \widetilde{s}_x \psi_{x,j}}$
$x = x^*$	$C^{\widehat{r} \widetilde{r}_{x,j}}$	$C^{\widetilde{r}_{x,j} \widehat{r} \tau d}$	DT	$D^{d_1^*} T^{d_2^*}$	$e(g,g)^{\phi_j f_x} e(g,g)^{d_1^2 \phi_j f_x}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$g^{d \widetilde{s}_x \tau \nu_j}$	g^{e_x}	$g^{d_1 d_x}$	$e(g,g)^{f_x \phi_j} e(g,g)^{d_1^2 \phi_j f_x}$

Table 34: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$\forall\, y \in [\widetilde{n}]$	$B^{c_{y,\ell,b}} h^{\widetilde{w}_{y,\ell,b} \tau} \mathfrak{g}_p^{\widetilde{v}_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$

Analysis of simulation. For $T = \mathfrak{g}_q^{a^2 c}$, $\mathscr{B}$ simulates the hybrid 4, otherwise if T is randomly chosen from the group $\mathbb{G}_q$ then $\mathscr{B}$ simulates the view of hybrid 5 as the target row '$= x^*$' will be indistinguishable from the less than row '$< x^*$'. Therefore, if $\mathscr{B}$ breaks the R3DH assumption with the advantage $\epsilon(\cdot)$ then $\mathscr{A}$ wins the game with the same advantages.

Hybrid 5 ≈ Hybrid 6: To prove the hybrid 5 and 6 are indistinguishable, let us consider $(2\widetilde{n}k+1)$ sub-hybrid $H_0, H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ for $(\widetilde{y}, \widetilde{\ell}, \widetilde{b}) \in ([\widetilde{n}] \times [k] \times \{0,1\})$. In this game the sub-hybrid H_0 corresponds to the hybrid 5 as described above. Now the sub-hybrid $H_{y,\ell,b}$ is same as the hybrid H_0 except that the column components in the challenge ciphertext $C_{y,\ell,b}$ for $y < \widetilde{y}$ and $(y, \ell, b) \in \{\widetilde{y}\} \times [\widetilde{\ell} - 1] \times \{0,1\}$ and for $y = \widetilde{y}, \ell = \widetilde{\ell}, b < \widetilde{b}$ have a random element in the subgroup $\mathbb{G}_p$. The column components are generated as described below in the Table 35.

Table 35: Computing column components of the ciphertext in sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell > \widetilde{\ell}) \vee$ $(y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b \geq \widetilde{b})$	$H_{y,\ell,b}^{t} \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell < \widetilde{\ell}) \vee$ $(y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$H_{y,\ell,b}^{t} \cdot h^{w_{y,\ell,b}\tau} \cdot V_{\ell,b}^{v_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$

In the following, we show that the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ is indistinguishability with the sub-hybrid $H_{\widetilde{y},\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$. Note that, the sub-hybrid $H_{1,1,0}$ is identical to the main hybrid 6. It is also required that the sub-hybrid $H_0 \approx H_{\widetilde{n},\ell,1}$ and sub-hybrid $H_{\widetilde{y},k,1} \approx H_{\widetilde{y}+1,1,0}$. We now show the here the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ and $H_{\widetilde{y},\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ are indistinguishable whose techniques are used to prove the indistinguishability of all the consecutive sub-hybrids for $(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}$. By combining all the proofs of sub-hybrids our required hybrids 5 and 6 will be indistinguishable.

Sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}} \approx$ *Sub-hybrid* $H_{\widetilde{y},\widetilde{y}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$: Suppose on the contrary, there exist a PPT adversary $\mathscr{A}$ that distinguishes the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$ and sub-hybrid $H_{\widetilde{y},\widetilde{y}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ with the non-negligible advantage $\epsilon(\cdot)$. We construct a PPT reduction algorithm $\mathscr{B}$ which can break the modified-2 D3DH assumption 3 with same non-negligible advantage as described above. From the challenger, the reduction algorithm $\mathscr{B}$ receives the following instances as

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, T)$$

where T is either g_p^{abc} or a random element in the subgroup $\mathbb{G}_p$ of prime order p. Next, it receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^{k'}, 1^m, x^*, y^*, \mathsf{gid}^*)$ from the adversary $\mathscr{A}$ where $y^* = \widetilde{n}$. Now, $\mathscr{B}$ generates the master public key using the modified-2 D3DH instances and sends it to the adversary. Then the adversary can adaptively make queries for the secret keys of distinct indices except the pair (i^*, gid^*). In the following, $\mathscr{B}$ generates the master public key, secret keys and the challenge ciphertext. Note that, finally $\mathscr{A}$ outputs its guess, which $\mathscr{B}$ uses to break the given modified-2 D3DH assumption 3. Since the reduction plays with the challenger over the subgroup $\mathbb{G}_p$, thus it can choose any elements from the subgroup $\mathbb{G}_q$ itself. It now implicitly sets $t_p = a \cdot b$, $h_p = B = g_p^b$ and $c_{p,\widetilde{y},\widetilde{\ell},\widetilde{b}} = c \cdot \widetilde{c}_{p,\widetilde{y},\widetilde{\ell},\widetilde{b}}$ where the exponent $\widetilde{c}_{p,\widetilde{y},\widetilde{\ell},\widetilde{b}}$ is chosen uniformly random. By using these exponents, $\mathscr{B}$ simulates the master public key, secret keys and the group elements T can be programmed in the challenge ciphertext components $C_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$.

Public key simulation. To generate the master public key, $\mathscr{B}$ sets $h_q = g_q^d$ for $d \in \mathbb{Z}_N$ and chooses the random exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, \quad & \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad & \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}], \quad & r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

Also, it chooses random group elements $\vartheta'_p, \vartheta_{p,i} \in \mathbb{G}_p$ for all $i \in [k']$ and $\vartheta'_q, \vartheta_{q,i} \in \mathbb{G}_q$ for all $i \in [k']$ such that $\mathrm{H}(\mathsf{gid}) = (\vartheta'_p \vartheta'_q) \prod_{i \in \mathcal{V}} \vartheta_{p,i} \vartheta_{q,i} = \vartheta' \prod_{i \in \mathcal{V}} \vartheta_i$ and $\mathrm{H}_q(\mathsf{gid}) = \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{q,i}$ corresponding to any group identity gid. It also samples $\beta, \widehat{r}, d_1, d_2 \leftarrow \mathbb{Z}_N$ and sets $g = g_p g_q$, $h = h_p h_q = B h_q = g_p^b \cdot g_q^d$, $f = f_p f_q = g_p^{d_1} g_q^{d_2}$. All the components of the master public keys

are generated as follows:

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = B h_q, f = g^d, \\ \vartheta', \vartheta_q'^{\beta}, \boldsymbol{\vartheta} = (\vartheta_i), \{\vartheta_{q,i}^{\beta}\}_{i\in[k']}, \mathrm{H}, \mathrm{H}_q, E_q = g_q^{\beta} \\ \left\{ \begin{array}{c} E_{x,j} = g^{\widehat{r} r_{x,j}}, F_{x,j} = h^{\widehat{r} r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}} \\ E_{q,x,j} = g_q^{\beta \widehat{r} r_{x,j}}, F_{q,x,j} = h_q^{\beta \widehat{r} r_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}}, \\ Y_{x,j} = g^{\psi_{x,j}}, Y_{q,x,j} = g_q^{\beta \psi_{x,j}} \end{array} \right\}_{x\in[\widehat{n}], j\in[m]} \\ \{H_{y,\ell,b}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b)\in[k]\times\{0,1\}} \end{array} \right)$$

All the components are generated from the challenge D3DH instances.

Secret key simulation. In the query phase, the adversary $\mathscr{A}$ is not allowed to query for the challenge index i^* and the group identity gid^* together. If $\mathsf{gid} = \mathsf{gid}^*$, $\mathscr{B}$ replies the secret key to $\mathscr{A}$ corresponding the index $i(\neq i^*)$ as

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \widetilde{\ell}} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} (C g_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{\widetilde{y},\widetilde{\ell},\widetilde{b}}} & \text{if } (y, \mathsf{id}_{\widetilde{\ell}}) = (\widetilde{y}, \widetilde{b}), \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{otherwise.} \end{cases},$$

$$K_2 = g^{d\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathsf{gid})^r, \;\; K_3 = g^r$$

where $\widetilde{r} \leftarrow \mathbb{Z}_N$ and set $r = \widetilde{r} \cdot \widehat{r}$. In the secret key query phase, adversary $\mathscr{A}$ is not allowed to secret key query for the index position $i^* = (x^*, y^*)$. If $\mathsf{gid} \neq \mathsf{gid}^*$, then $\mathscr{A}$ can query for the secret for the index position i^*. Then the secret keys are generated as

$$K_1 = \begin{cases} g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \widetilde{\ell}} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} (C g_q)^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{\widetilde{y},\widetilde{\ell},\widetilde{b}}} & \text{if } y = \widetilde{y} = \widetilde{n} \wedge \mathsf{id}_{\widetilde{\ell}} = \widetilde{b}, \\ g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} & \text{if } y = \widetilde{y} = \widetilde{n} \wedge \mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b}, \end{cases}$$

To simulate the other keys components, the challenger similarly construct the function F, J and K functions as mentioned in hybrid 2 to hybrid 3 secret key simulation phase. Also, $\mathscr{B}$ sets similar public key parameters ϑ' and ϑ_i's. It answers the remaining secret key components as follows:

$$\begin{aligned} K_2 &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \mathrm{H}(\mathsf{gid})^r \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r'} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathsf{gid})^{r'} \\ K_3 &= g^r \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r'} \end{aligned}$$

As in the previous argument, we implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}$. Therefore, from the construction of K function, it implies that $\mathrm{K}(\mathsf{gid}) \neq 0$ for any key query corresponding to the group identity gid. This implies the function $\mathrm{F}(\mathsf{gid}) \neq 0 \mod N$ for any identities (since we assume $N > sk_1'$ for any reasonable value of N, s and k_1'). To prove this, we need the following Lemma 16.

Challenge ciphertext simulation. Challenger $\mathscr{B}$ chooses the random exponents as follows.

$$\begin{aligned} \forall\, j \in [m], \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N,\ \tau \in \mathbb{Z}_N, t_q \leftarrow \mathbb{Z}_N \\ \forall\, x \in [\widehat{n}], \quad e_x, f_x, d_x, s_x \leftarrow \mathbb{Z}_N \\ \forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad \widetilde{w}_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N \end{aligned}$$

For the challenge identity gid^*, $\mathscr{B}$ computes $\mathrm{H}(\mathsf{gid}^*) = (\vartheta'_p \vartheta'_q) \prod_{i \in \mathcal{V}^*} \vartheta_{p,i} \vartheta_{q,i} = g_p^{d_1^*} g_q^{d_2^*}$ and $\mathrm{H}_q(\mathsf{gid}^*)^\beta = \vartheta_q^{'\beta} \prod_{i \in \mathcal{V}^*} \vartheta_{q,i}^\beta = g_q^{\beta d_2^*}$ for some $d_1^*, d_2^* \in \mathbb{Z}_N$. Now, for all $x \in [\widehat{n}]$, $j \in [m]$, the challenger can compute following row and column components.

Table 36: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$E_q^{s_x t_q}$	$\mathrm{H}_q(\mathsf{gid}^*)^{\beta s_x t_q}$	$e(g_q, g_q)^{\nu_j} \cdot G_{q,x,j}^{s_x t_q} \cdot e(f_q, Y_{q,x,j})^{s_x t_q}$
$x \le x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	$\mathrm{H}(\mathsf{gid})^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 37: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell > \widetilde{\ell}) \vee$ $(y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\widetilde{w}_{y,\ell,b} \tau}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} \cdot g^{\widetilde{w}_{y,\ell,b}}$
$y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b = \widetilde{b}$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\tau \widetilde{w}_{y,\ell,b}} T^{\widetilde{c}_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$
$(y < \widetilde{y}) \vee (y = \widetilde{y} \wedge \ell < \widetilde{\ell}) \vee$ $(y = \widetilde{y} \wedge \ell = \widetilde{\ell} \wedge b < \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\tau \widetilde{w}_{y,\ell,b}} g_p^{\upsilon_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$

Analysis of simulation. For $T = g_p^{abc}$, then $\mathscr{A}$ gets the view of the challenge ciphertext as the sub-hybrid $H_{\widetilde{y},\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$, otherwise for any random group elements from the subgroup $\mathbb{G}_p$, the adversary $\mathscr{A}$ gets the view of the sub-hybrid $H_{\widetilde{y},\widetilde{\ell},\widetilde{b}}$. Therefore, if $\mathscr{A}$ breaks the modified D3DH assumption 1 with non-negligible advantage then it also wins the game with the same advantage.

Hybrid 6 ≈ Hybrid 7: Suppose on the contrary, there exists a PPT adversary $\mathscr{A}$ that can distinguish between the hybrid 6 and hybrid 7 with non-negligible advantage $\epsilon(\cdot)$. Now, we construct a PPT reduction algorithm $\mathscr{B}$ that breaks the DHSD assumption 4 with the same advantage.

The reduction algorithm $\mathscr{B}$ first receives the DHSD challenge instances of assumption 4 from its challenger as

$$(\mathbb{BG}, g = g_p g_q, h = h_p h_q, A = g_q^a, B = h_q^a, C = g^b g_p^c, D = h^b, T)$$

where T is either sampled as $T = g_q^d$ or $T = g^d$, where d is a random exponent sampled as $d \leftarrow \mathbb{Z}_N$. Next $\mathscr{B}$ receives the challenge tuple $(1^\lambda, 1^n, 1^k, 1^{k'}, 1^m, x^*, y^*, \mathsf{gid}^*)$ from the adversary for $y^* = \widetilde{n}$. Now, $\mathscr{B}$ generates the master public key from the instances of assumption 4 and sends it to the adversary. Then, adversary is not allowed to make adaptively secret key queries for the tuple $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$. Now, $\mathscr{B}$ simulates the secret keys and the challenge ciphertext using the instances of DHSD assumption 4. Finally, $\mathscr{A}$ outputs its guess, which $\mathscr{B}$ uses to break the assumption 4. In this proof, the reduction plays with its challenger in the

subgroup $\mathbb{G}_p$ thus it can choose any components from the subgroup $\mathbb{G}_q$ by itself. Let us consider, $\mathscr{B}$ first implicitly sets the random exponents as $\beta = a, s_{x^*+1} = d \cdot \widetilde{s}_{x^*+1}, \gamma_{\ell,b} = c \cdot \widetilde{\gamma}_{\ell,b}$ and $\delta_{\ell,b} = b \cdot \widetilde{\gamma}_{\ell,b} + \widetilde{\delta}_{\ell,b}$ where the exponents $\widetilde{\gamma}_{\ell,b}, \widetilde{\delta}_{\ell,b}$ are uniformly chosen at random. Also, $\mathscr{B}$ implicitly sets $h^{\tau} = g^{\pi}$ where π be any random exponent from $\mathbb{Z}_N$. In this simulation, the challenge group element T can be programmed in the challenge ciphertext to compute the row components for $x = x^* + 1$.

Public key simulation. Challenger $\mathscr{B}$ chooses the random exponents as follows:

$$\begin{aligned} \forall\, \ell \in [k], b \in \{0,1\}, \quad & \widetilde{\delta}_{\ell,b} \leftarrow \mathbb{Z}_N, \widetilde{\gamma}_{\ell,b} \leftarrow \mathbb{Z}_p, \\ \forall\, y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \quad & c_{y,\ell,b} \leftarrow \mathbb{Z}_N, \\ \forall\, j \in [m], x \in [\widehat{n}], \quad & r_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N. \end{aligned}$$

It samples $d', \widehat{r} \leftarrow \mathbb{Z}_N$ such that $f_q = g_q^{d'}$. It also samples random group elements $\vartheta'_p, \vartheta_{p,i} \in \mathbb{G}_p$ for all $i \in [k']$ and $\vartheta'_q, \vartheta_{q,i} \in \mathbb{G}_q$ for all $i \in [k']$ such that $\mathrm{H}(\mathsf{gid})^{\beta} = (\vartheta'_p \vartheta'_q) \prod_{i \in \mathcal{V}} \vartheta_{p,i} \vartheta_{q,i} = \vartheta' \prod_{i \in \mathcal{V}} \vartheta_i$ and $\mathrm{H}_q(\mathsf{gid}) = \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{q,i}$ corresponding any group identity gid. We write $\vartheta_q^{'\beta} = g_q^{a\varrho} = A^{\varrho}$, $\vartheta_{q,i}^{\beta} = g_q^{a\varrho_i} = A^{\varrho_i}$ where $\varrho, \varrho_i \leftarrow \mathbb{Z}_q$ In the following, $\mathscr{B}$ computes the master public components using the challenge instances of assumption 4 as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g, h, f, \vartheta', \vartheta_q^{'\beta}, \boldsymbol{\vartheta} = (\vartheta)_i, \{\vartheta_{q,i}^{\beta}\}_{i \in [k']}, \mathrm{H}, \mathrm{H}_q, E_q = A, \\ \left\{ \begin{array}{c} E_{x,j} = g^{\widehat{r} r_{x,j}}, F_{x,j} = h^{\widehat{r} r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}}, \\ E_{q,x,j} = A^{\widehat{r} r_{x,j}}, F_{q,x,j} = B^{\widehat{r} r_{x,j}}, G_{q,x,j} = e(A, g_q)^{\alpha_{x,j}}, \\ Y_{x,j} = g^{\psi_{x,j}}, Y_{q,x,j} = A^{\psi_{x,j}} \end{array} \right\}_{x \in [\widehat{n}], j \in [m]}, \\ \{H_{y,\ell,b} = g^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\widetilde{\delta}_{\ell,b}} C^{\widetilde{\gamma}_{\ell,b}}, V_{\ell,b} = h^{\widetilde{\delta}_{\ell,b}} D^{\widetilde{\gamma}_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right),$$

Secret Key Simulation. The challenger $\mathscr{B}$ answers the adversary's queried secret keys corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$.

If $\mathsf{gid} = \mathsf{gid}^*$, the adversary can not make any secret key queries for the index i^*. Therefore, the secret keys corresponding to the index $i (\neq i^*)$ are generated as follows:

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\langle \boldsymbol{r}_x, \boldsymbol{u} \rangle}, \quad K_2 = f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathsf{gid})^r, \quad K_3 = g^r$$

where $\widetilde{r}$ is randomly sampled from $\mathbb{Z}_N$ such that $r = \widetilde{r} \cdot \widehat{r}$. Note that, in this case the adversary can not query for the secret key corresponding to the index i^*.

If $\mathsf{gid} \neq \mathsf{gid}^*$, the adversary can make secret key query index position i^*. In that case, $\mathscr{B}$ answers the secret key $\mathsf{sk}_{\boldsymbol{u}}$ as follows

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\widehat{r} \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle} \quad \text{if } x = x^*, y = y^*$$

The other secret keys are similarly generated as the previous hybrid. For $\mathsf{gid} \neq \mathsf{gid}^*$, challenger construct the identity encoding functions F, J and K similarly. Now, $\mathscr{B}$ answers the remaining secret keys components K_2, K_3 looks as follows:

$$K_2 = g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \mathrm{H}(\mathsf{gid})^r$$

$$
\begin{aligned}
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})} \right)^{r'} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \mathrm{H}(\mathsf{gid})^{r'} \\
K_3 = g^r \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} &= g^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r'}
\end{aligned}
$$

As the similar argument, we can implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}$.

Challenge ciphertext simulation. $\mathscr{B}$ chooses the random exponents as follows:

$$
\begin{aligned}
\forall j \in [m], \;\; \sigma_j, \nu_j, \phi_j \leftarrow \mathbb{Z}_N, \;\; \pi, t \leftarrow \mathbb{Z}_N, \\
\forall x \in [\widehat{n}], \;\; e_x, f_x, \widetilde{s}_x, d_x \leftarrow \mathbb{Z}_N \\
\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\}, \;\; w_{y,\ell,b}, \upsilon_{y,\ell,b} \leftarrow \mathbb{Z}_N
\end{aligned}
$$

For the challenge identity gid^*, $\mathrm{H}(\mathsf{gid}^*) = (\vartheta'_p \vartheta'_q) \prod_{i \in \mathcal{V}^*} \vartheta_{p,i} \vartheta_{q,i} = g_p^{d_1^*} g_q^{d_2^*} = g^{d^*}$ and $\mathrm{H}_q(\mathsf{gid}^*) = \vartheta'_q \prod_{i \in \mathcal{V}^*} \vartheta_{q,i} = g_q^{d_2^*}$ for some $d^*, d_1^*, d_2^* \leftarrow \mathbb{Z}_N$. So, $\mathrm{H}_q(\mathsf{gid}^*)^\beta = A^{d_2^*}$, Now, row and columns components of the challenge ciphertext are generated as in Table 38, 39.

Table 38: Computing row components of the ciphertext for $x \in [\widehat{n}]$, $j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^* + 1$	$E_{q,x,j}^{\widetilde{s}_x}$	$F_{q,x,j}^{\widetilde{s}_x \pi}$	$E_q^{\widetilde{s}_x t}$	$A^{d_2^* t_q \widetilde{s}_x}$	$e(g_q, g_q)^{\upsilon_j} \cdot G_{q,x,j}^{\widetilde{s}_x t} \cdot e(f_q, A)^{t \widetilde{s}_x \psi_{x,j}}$
$x = x^* + 1$	$T^{\widetilde{s}_x r_{x,j}}$	$T^{\widetilde{s}_x r_{x,j} \pi}$	$T^{\widetilde{s}_x t}$	$T^{d^* t \widetilde{s}_x}$	$e(g,g)^{\upsilon_j} \cdot e(T,g)^{\widetilde{s}_x \alpha_{x,j} t} \cdot e(f,T)^{t \psi_{x,j} s_x}$
$x < x^* + 1$	$g^{\widetilde{s}_x \sigma_j}$	$h^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	$\mathrm{H}(\mathsf{gid}^*)^{d_x}$	$e(g,g)^{f_x \phi_j} e(f,f)^{f_x \phi_j}$

Table 39: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$\forall (y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$	$H_{y,\ell,b}^t \cdot g^{w_{y,\ell,b} \pi}$	$g^{w_{y,\ell,b}}$

After seeing the challenge ciphertext, $\mathscr{B}$ receives a guess bit $\mathfrak{b}'$ from $\mathscr{A}$ and it simply forwards that as its guess to the challenger of assumption 4.

Analysis of simulation. Finally, if $T = g_q^d$, then $\mathscr{B}$ simulates the view of the hybrid 6, otherwise, if T is randomly chosen element from the group $\mathbb{G}$ then $\mathscr{B}$ simulates the view same as hybrid 7. Therefore, if $\mathscr{A}$ wins with the non-negligible advantages $\epsilon(\cdot)$ then $\mathscr{B}$ breaks the DHSD assumption 4 with the same advantage. Hence proof of the Lemma 9 is complete. □

This concludes the proof of index-hiding security. □

Lemma 13 (Lower identity-hiding) *If the modified D3DH assumption 1 holds over the bilinear group $\mathbb{BG}$, then our EIPL-IBIPFE satisfies the lower identity-hiding security as per the Definition 15.*

Proof. We recall that in the lower identity-hiding security it is required that no PPT adversary can distinguish between the special encryption to the index-position-bit tuple ($i^* =$

$(x^*, y^*), \ell^*, b^*)$ and $(i^* = (x^*, y^*), \perp, 0)$ with non-negligible advantages. In its key query phase, the adversary is not allowed to secret key query for the tuple $(i^* = (x^*, y^*), \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = b^*$. This proof technique is nearly identical to that of Lemma 8. Here, we just exclude the intermediate hybrids as mentioned in the previous proof. Let $(i^* = (x^*, y^*), \ell^*, b^*)$ be the challenge tuple provided by the adversary $\mathscr{A}$. Then the hybrid H_{ℓ^*, b^*} corresponds to the exactly same as the lower identity-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \ell^*, b^*)$ and similarly the hybrid $H_{0,1}$ is same as the lower identity-hiding game for the index-position-bit tuple $(i^* = (x^*, y^*), \perp, 0)$. So the indistinguishability proof to the tuple $(i^* = (x^*, y^*), \perp, 0)$ and $(i^* = (x^*, y^*), \ell^*, b^*)$ are similar to the Lemma 8.
In the following, we discuss the secret key simulation where the reduction algorithm $\mathscr{B}$ answers all permissible secret keys corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ as per the lower identity-hiding security game. From the security restriction of this game, the adversary can not query for the secret key corresponds to the tuple $(i^* = (x^*, y^*), \mathsf{id}_{\ell^*} = b^*, \mathsf{gid}^*, \boldsymbol{u})$. So all the key queries are of the form either $i \neq i^*$ or $\mathsf{id}_{\ell^*} \neq b^*$ or $\mathsf{gid} \neq \mathsf{gid}^*$.

Secret key simulation. To answer the secret key corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$, $\mathscr{B}$ samples a random value $\widetilde{r} \leftarrow \mathbb{Z}_N$ and sets $r = \widetilde{r} \cdot \widehat{r}$ and it computes $\mathrm{H}(\mathsf{gid}^*) = g_p^{d_1^*} g_q^{d_2^*}$ for some $d_1^*, d_2^* \leftarrow \mathbb{Z}_N$. Note that, the adversary is not allowed to query for the secret key corresponding to the tuple $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = b^*$.

If $\mathsf{gid} = \mathsf{gid}^*$, $\mathscr{A}$ can not query for the secret key corresponding to the index i^* such that $\mathsf{id}^* = b^*$. In the following Table 40, $\mathscr{B}$ generates all possible keys corresponding the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$.

conditions	sub-conditions on (x, y)	secret keys		
		K_1	K_2	K_3
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x \neq x^* \wedge y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x \neq x^* \wedge y = y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (E g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,b^*}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x = x^* \wedge y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (F g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (F g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,b^*}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i = i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x = x^*) \wedge (y = y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (F g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i = i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x = x^*) \wedge (y = y^*)$	$g_q^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle + \widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle} \cdot g_p^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot F^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot H^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,b^*}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x \neq x^* \wedge y \neq y^*) \wedge$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x = x^* \wedge y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (F g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x \neq x^* \wedge y = y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,\mathsf{id}_{\ell^*}}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(D g_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i = i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x = x^*) \wedge (y = y^*)$	$g_q^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle + \widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle} \cdot g_p^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot F^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x \neq x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x = x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (F g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x \neq x^*) \wedge (y = y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (E g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,b^*}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x \neq x^*) \wedge (y = y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \neq \ell^*} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \widetilde{c}_{y,\ell^*,\mathsf{id}_{\ell^*}}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x \neq x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (D g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x = x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (F g_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$

Table 40: Simulated secret keys used in the lower identity-hiding game

For $\mathsf{gid} \neq \mathsf{gid}^*$, $\mathscr{B}$ generates the K_2, K_3 secret key components as follows:

We assume that the adversary makes the maximum number of Q queries and the challenge group identity gid^* and challenge index i^*. Now, the simulator chooses an integer $k_1' \leftarrow [k']$, sets an integer $s = 10Q$, a random k'-length vector $\boldsymbol{z} = (z_i) \leftarrow \mathbb{Z}_s^{k'}$ and a value $z' \leftarrow \mathbb{Z}_s$. Additionally, the simulator also chooses a random value $w' \leftarrow \mathbb{Z}_N$ and a uniformly random k'-length vector $\boldsymbol{w} = (w_i) \leftarrow \mathbb{Z}_N^{k'}$. All these values are kept secret to the simulator.

Let us consider $\mathcal{V}^* \subseteq \{1,2,\ldots,k'\}$ be the set of all i for which the challenge identity $\mathsf{gid}_i^* = 1$. Let $\mathcal{V}^* = \{i_1, i_2, \ldots, i_\kappa\}$. Now, we choose the z_i values from $\boldsymbol{z}$ which correspond to the collection of indices $\mathcal{V}^*$. Then set $\sum_{i\in\mathcal{V}^*} z_i = k_1' s - z'$ for uniformly chosen $k_1' \in [k']$. Now, we define the function K(gid) as

$$\mathrm{K}(\mathsf{gid}) = \begin{cases} 0, & \text{if } z' + \sum_{i\in\mathcal{V}} z_i \equiv 0 \mod s \\ 1, & \text{elsewhere} \end{cases}$$

So, from the above definition of the function K, we can say that $\mathrm{K}(\mathsf{gid}^*) = 0$ and for all $\mathsf{gid}(\neq \mathsf{gid}^*)$ it becomes non-zero. Additionally, we set two functions as $\mathrm{F}(\mathsf{gid}) = N - sk_1' + z' + \sum_{i\in\mathcal{V}} z_i$ and $\mathrm{J}(\mathsf{gid}) = w' + \sum_{i\in\mathcal{V}} w_i$. The simulator assigns the public parameters $\vartheta' = f^{N-k_1's+z'} \cdot g^{w'} = g_p^{d_p'} g_q^{d_q'}$ and $\vartheta_i = f^{z_i} g^{w_i} = g_p^{d_{p,i}} g_q^{d_{q,i}}$. Now $\mathscr{B}$ answers remaining secret key components as

$$\begin{aligned}
K_2 &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \tilde{r}} g_q^{d_2^* \hat{r}_q \tilde{r}} \\
&= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \mathrm{H}(\mathsf{gid})^r \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^r \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r'} \\
&= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid})^{r'} \\
K_3 &= (D^{\tilde{r}} g_q^{\hat{r}_q \tilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r'}
\end{aligned}$$

We implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}$. So from the construction of K function, we get $\mathrm{K}(\mathsf{gid}) \neq 0$ for any key query corresponding to the group identity gid. This implies that the function $\mathrm{F}(\mathsf{gid}) \neq 0 \mod N$ for any identity (since we assume $N > sk_1'$ for any reasonable value of N, s and k_1'). To prove this, we need the following Lemma 16.

Thus, the above shows that the reduction algorithm can perfectly simulate the lower identity-hiding game, thereby implying that the scheme satisfies lower identity-hiding security, assuming that the modified D3DH assumption 1 holds.

Lemma 14 (Upper identity-hiding) *If the assumptions 1,4,5 and 6 hold over the bilinear group $\mathbb{BG}$, then our EIPL-IBIPFE satisfies the upper identity-hiding security as per the Definition 15.*

Proof. The upper identity-hiding security requires that no PPT adversary can distinguish between the special encryption to the index-position-bit tuple (i^*, ℓ^*, b^*) and $(i^*+1, \perp, 0)$ with a non-negligible advantage. In the security experiment, the adversary makes only one secret key query for some index position and it is not allowed to make only key query for the tuple $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} = 1 - b^*$. To prove the Lemma 14, we consider a sequence of hybrid games as discuss below.

$\mathbf{Hyb_1}$: The hybrid corresponding to the upper identity-hiding game in which the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^*), \ell^*, b^*)$.

$\mathbf{Hyb_2}$: The hybrid is the similar as the Hyb_1 except that the column components as the table below.

Table 41: Computing column components for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell \neq \ell^*)$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee ((y,\ell) = (y^*, \ell^*))$	$H^t_{y,\ell,b} \cdot h^{w_{y,\ell,b}\tau} \cdot V^{\tau v_{y,\ell,b}}_{\ell,b}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}^{v_{y,\ell,b}}_{\ell,b}$

In words, we can say that the ciphertext component $C_{y^*,\ell^*,1-b^*}$ also includes random elements from the subgroup $\mathbb{G}_p$ whereas in Hyb_1 only C_{y^*,ℓ^*,b^*} for the index position i^* include a random element from the subgroup $\mathbb{G}_p$.

$\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ where $(\widetilde{\ell}, \widetilde{b}) \in [k] \times \{0,1\}$: This hybrid is same as Hyb_2 except that the column components in the challenge ciphertext are computed as the following Table 42. In words, we can say that the challenge ciphertext ciphertext components $C_{y^*,\ell,b}$ for $\ell < \widetilde{\ell}$, or $\ell = \widetilde{\ell}$ and $b \leq \widetilde{b}$ include a random component from the subgroup $\mathbb{G}_p$.

Table 42: Computing column components for the sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee$ $(y = y^* \wedge \ell \notin [\widetilde{\ell}] \cup \{\ell^*\}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$H^t_{y,\ell,b} h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee$ $(y = y^* \wedge \ell \in [\widetilde{\ell} - 1] \cup \{\ell^*\}) \vee$ $(y = y^* \wedge \ell = \widetilde{\ell} \wedge b \leq \widetilde{b})$	$H^t_{y,\ell,b} h^{w_{y,\ell,b}\tau} V^{\tau v_{y,\ell,b}}_{\ell,b}$	$g^{w_{y,\ell,b}}$

Hyb_4 : This hybrid is similar to the previous sub-hybrid $\mathsf{Hyb}_{3,k,1}$ except that the challenge ciphertext is a special encryption to the index-position-bit tuple $(i^* = (x^*, y^* + 1), \perp, 0)$. Note that, here $y^* = \widetilde{n}$ could be equal to $\widetilde{n}$. In that case, the special-encryption algorithm can be directly extended to encrypt such position.

Hyb_5. This hybrid corresponds to the upper identity-hiding game in which the challenge ciphertext is an encryption to the index-position-bit tuple $(i^* + 1, \perp, 0)$. Note that if $y^* \neq \widetilde{n}$, then the hybrids 4 and 5 are already identical.

Next, we discuss the indistinguishability of the above hybrids.

$\mathsf{Hyb}_1 \approx \mathsf{Hyb}_2$: The indistinguishability proof of the hybrids Hyb_1 and Hyb_2 is identical to that of Claim 8 and Lemma 13.

$\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}} \approx \mathsf{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$: If the modified-1 D3DH assumption 1 holds, there does not exists any PPT adversary that can distinguish between the sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ and the sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$ with non-negligible advantage. For, $\widetilde{\ell} = \ell^*$, then sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ is identically equals to sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,(\widetilde{b}+1) \mod 2}$. Otherwise, according to the key query there have two cases.

Case 1. Adversary makes a key query for index tuple $(j, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ such that $j = i^* \wedge \mathsf{id}_{\widetilde{\ell}} =$

$\widetilde{b} \wedge \mathsf{gid} = \mathsf{gid}^*$.

Suppose on contrary, there exists a PPT adversary $\mathscr{A}$ which can distinguish between the sub-hybrids $\mathrm{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$ and $\mathrm{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,\widetilde{b}+1 \mod 2}$ with non-negligible probability. We construct a PPT reduction algorithm $\mathscr{B}$ which breaks the assumption 1 with same non-negligible advantage as

$$(\mathbb{BG}, g_p, g_q, A = g_p^a, B = g_p^b, C = g_p^c, D = g_p^{b^2}, E = g_p^{b^2 c}, F = g_p^{b^3}, G = g_p^{b^4}, H = g_p^{b^3 c}, T)$$

where T is either g_p^{abc} or an uniformly random element from the sub-group $\mathbb{G}_p$. Next, $\mathscr{B}$ receives a challenge tuple $(1^\lambda, 1^n, 1^k, 1^{k'}, 1^m, (i^*, \ell^*, b^*))$ from $\mathscr{A}$. Then, $\mathscr{B}$ generates the master public key and sends it to $\mathscr{A}$. After seeing mpk, the adversary makes polynomial numbers of secret keys for the distinct index positions i under some admissible conditions. Then $\mathscr{B}$ simulates the public keys, secret keys and the challenge ciphertext and sends it to $\mathscr{A}$. Finally, the adversary outputs a bit $\mathfrak{b}'$ as guess which $\mathscr{B}$ uses to breaks the assumption 1. As the reduction plays the game with its challenger in the subgroup $\mathbb{G}_p$ so everything it can choose from the subgroup $\mathbb{G}_q$ by itself. Let us implicitly set the exponents as below

$$t_p = ab;\ \ r_{p,x^*,j} = b \cdot \widetilde{r}_{p,x^*,j};\ \ c_{p,y^*,\ell^*,b^*} = c + \widetilde{c}_{p,y^*,\ell^*,b^*};$$
$$\widehat{r}_p = b^2,\ s_{p,x^*} = \widetilde{s}_{p,x^*}/b;\ \ c_{p,y^*,\widetilde{\ell},\widetilde{b}} = -c + \widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}}$$

where $\widetilde{r}_{p,x^*,j}, \widetilde{s}_{p,x^*}, \widetilde{c}_{p,y^*,\widetilde{\ell},\widetilde{b}}$ and $\widetilde{c}_{p,y^*,\ell^*,b^*}$ are the random exponents. Also, we implicitly set $h_p = B = g_p^b$ and $f_p = B^{d_1}$ for d_1 uniformly chosen from $\mathbb{Z}_N$. Setting the exponents allows to simulate the public key, secret key exactly as well as the challenge group elements T, that can be programmed in the challenge ciphertext components, $C_{y^*,\widetilde{\ell},\widetilde{b}}$.

Public key simulation. The challenger $\mathscr{B}$ chooses a random group generator $h_q \in \mathbb{G}_q$ by sampling random exponent $d \in \mathbb{Z}_N$ such that $h_q = g_q^d$. To generate the public key, $\mathscr{B}$ additionally chooses the following exponents

$$\forall \ell \in [k], b \in \{0,1\},\ \ \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p,$$
$$\forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\},\ \ \widetilde{c}_{y,\ell,b} \leftarrow \mathbb{Z}_N,$$
$$\forall j \in [m], x \in [\widehat{n}],\ \ \widetilde{r}_{x,j}, \alpha_{x,j}, \psi_{x,j} \leftarrow \mathbb{Z}_N.$$

It samples $\beta, \widehat{r}_q \leftarrow \mathbb{Z}_N$ and chooses the random generator f_q from the subgroup $\mathbb{G}_q$ such that $f_q = g_q^{d'}$ for some $d' \in \mathbb{Z}_N$. Then, $\mathscr{B}$ computes the components of the master public key by using the instances of modified-1 D3DH assumption 1.

$$E_{x,j} = \begin{cases} (D g_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} & \text{if } x \neq x^*, \\ (F g_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} & \text{elsewhere.} \end{cases}, \quad F_{x,j} = \begin{cases} (F h_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} & \text{if } x \neq x^* \\ (G h_q^{\widehat{r}_q})^{\widetilde{r}_{x,j}} & \text{elsewhere} \end{cases} \quad Y_{x,j} = g^{\psi_{x,j}}\ \forall x$$

$$H_{y,\ell,b} = \begin{cases} C g^{\widetilde{c}_{y,\ell,b}} & \text{if } (y,\ell,b) = (y^*,\ell^*,b^*) \\ C^{-1} g^{\widetilde{c}_{y,\ell,b}} & \text{if} (y,\ell,b) = (y^*,\widetilde{\ell},\widetilde{b}) \\ g^{\widetilde{c}_{y,\ell,b}} & \text{elsewhere} \end{cases}$$

The challenger also computes $E_{q,x,j} = g_q^{\beta \widehat{r}_q \widetilde{r}_{x,j}}, F_{q,x,j} = h_q^{\beta \widehat{r}_q \widetilde{r}_{x,j}}$ and $G_{x,j} = e(g,g)^{\alpha_{x,j}}$, $G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}}, Y_{q,x,j} = g_q^{\beta \psi_{x,j}}$. Then it samples the elements $\vartheta'_p, \vartheta_{p,i} \in \mathbb{G}_p$ for all $i \in [k']$, $\vartheta'_q, \vartheta_{q,i} \in \mathbb{G}_q$ for all $i \in [k']$ such that $\vartheta' = \vartheta'_p \vartheta'_q$ and $\boldsymbol{\vartheta} = (\vartheta_i) = (\vartheta_{p,i} \vartheta_{q,i})$. Let us, consider two identity encoding function H, H_q be defined as $\mathrm{H}(\mathsf{gid}) = \vartheta' \prod_{i \in \mathcal{V}} \vartheta_i$, $\mathrm{H}_q(\mathsf{gid}) = \vartheta'_q \prod_{i \in \mathcal{V}} \vartheta_{q,i}$. Finally, it publishes the master public key as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, g = g_p g_q, h = Bh_q, f = B^{d_1} f_q, \\ \vartheta', \vartheta_q'^{\beta}, \boldsymbol{\vartheta}, \{\vartheta_{q,i}^{\beta}\}_{i\in[k']}, \mathrm{H}, \mathrm{H}_q, E_q = g_q^{\beta}, \\ \left\{ \begin{array}{c} E_{x,j}, F_{x,j}, G_{x,j}, Y_{x,j}, \\ E_{q,x,j}, F_{q,x,j}, G_{q,x,j}, Y_{q,x,j} \end{array} \right\}_{x\in[\widehat{n}], j\in[m]} \\ \{H_{y,\ell,b}\}_{(y,\ell,b)\in[\widetilde{n}]\times[k]\times\{0,1\}}, \\ \left\{\widetilde{V}_{\ell,b}, V_{\ell,b}\right\}_{(\ell,b)\in[k]\times\{0,1\}} \end{array} \right)$$

Secret key simulation. To answer the secret key $\mathsf{sk}_{\boldsymbol{u}}$ corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$, $\mathscr{B}$ samples a random value $\widetilde{r} \leftarrow \mathbb{Z}_N$ and sets $r = \widetilde{r} \cdot \widehat{r}$ and for the challenge group identity gid^*, it computes $\mathrm{H}(\mathsf{gid}^*) = g_p^{d_1^*} g_q^{d_2^*}$ where $d_1^*, d_2^* \leftarrow \mathbb{Z}_N$. In the secret key query phase, $\mathscr{A}$ is not allowed to secret key query corresponding to the tuple $(i^*, \mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ such that $\mathsf{id}_{\ell^*} \neq b^*$. Now, $\mathscr{B}$ simulates the following secret keys in the Table 43 corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$.

To simulate the secret keys components K_2 and K_3 for the case $\mathsf{gid} \neq \mathsf{gid}^*$, we assume that the adversary makes the maximum number of Q queries and the challenge group identity gid^* and challenge index i^*. Now, the simulator chooses an integer $k_1' \leftarrow [k']$, sets an integer $s = 10Q$, a random k'-length vector $\boldsymbol{z} = (z_i) \leftarrow \mathbb{Z}_s^{k'}$ and a value $z' \leftarrow \mathbb{Z}_s$. Additionally, the simulator also chooses a random value $w' \leftarrow \mathbb{Z}_N$ and a uniformly random k'-length vector $\boldsymbol{w} = (w_i) \leftarrow \mathbb{Z}_N^{k'}$. All these values are kept secret to the simulator.

Let us consider $\mathcal{V}^* \subseteq \{1, 2, \ldots, k'\}$ be the set of all i for which the challenge identity $\mathsf{gid}_i^* = 1$. Let $\mathcal{V}^* = \{i_1, i_2, \ldots, i_\kappa\}$. Now, we choose the z_i values from $\boldsymbol{z}$ which correspond to the collection of indices $\mathcal{V}^*$. Then set $\sum_{i\in\mathcal{V}^*} z_i = k_1' s - z'$ for uniformly chosen $k_1' \in [k']$. Now, we define the function $\mathrm{K}(\mathsf{gid})$ as

$$\mathrm{K}(\mathsf{gid}) = \begin{cases} 0, & \text{if } z' + \sum_{i\in\mathcal{V}} z_i \equiv 0 \mod s \\ 1, & \text{elsewhere} \end{cases}$$

So, from the above definition of the function K, we can say that $\mathrm{K}(\mathsf{gid}^*) = 0$ and for all $\mathsf{gid}(\neq \mathsf{gid}^*)$ it becomes non-zero. Additionally, we set two functions as $\mathrm{F}(\mathsf{gid}) = N - sk_1' + z' + \sum_{i\in\mathcal{V}} z_i$ and $\mathrm{J}(\mathsf{gid}) = w' + \sum_{i\in\mathcal{V}} w_i$. The simulator assigns the public parameters $\vartheta' = f^{N-k_1's+z'} \cdot g^{w'} = g_p^{d_p'} g_q^{d_q'}$ and $\vartheta_i = f^{z_i} g^{w_i} = g_p^{d_{p,i}} g_q^{d_{q,i}}$. Now $\mathscr{B}$ answers remaining secret key components as

$$\begin{aligned} K_2 &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}} \\ &= g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \mathrm{H}(\mathsf{gid})^r \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^r \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid})} g^{\mathrm{J}(\mathsf{gid})}\right)^{r'} \\ &= f^{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid})^{r'} \\ K_3 &= (D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}} = g^{r'} \end{aligned}$$

We implicitly set $r' = r - \frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u}\rangle}{\mathrm{F}(\mathsf{gid})}$. So from the construction of K function, we get $\mathrm{K}(\mathsf{gid}) \neq 0$ for any key query corresponding to the group identity gid. This implies that the function

conditions	sub-conditions on (x, y) & $\mathsf{id}_{\widetilde{b}}$	secret keys		
		K_1	K_2	K_3
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x \neq x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x = x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (ED^{\sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x \neq x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x = x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot (E^{-1} D^{\sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i = i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x = x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i = i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$	$(x = x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$(B^{d_1} f_q)^{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle} \cdot (D^{d_1^*} g_q^{d_2^* \widehat{r}_q})^{\widetilde{r}}$	$(Dg_q^{\widehat{r}_q})^{\widetilde{r}}$
$(i = i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x = x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_q^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x = x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{\ell}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot [F^{\sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot H]^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} \neq b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x \neq x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_p^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x = x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_p^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{b}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot [D^{\sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot E^{-1}]^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{b}} \neq \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_p^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
$(i \neq i^*) \wedge (\mathsf{id}_{\ell^*} = b^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$	$(x \neq x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_p^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x = x^*) \wedge (y \neq y^*)$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Fg_p^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{b}} = \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \cdot (Dg_p^{\widehat{r}_q})^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$
	$(x \neq x^*) \wedge (y = y^*) \wedge (\mathsf{id}_{\widetilde{b}} \neq \widetilde{b})$	$g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} g_q^{\widehat{r}_q \langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle \sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot [D^{\sum_{\ell \in [k]} \widetilde{c}_{y,\ell,\mathsf{id}_\ell}} \cdot E]^{\langle \widetilde{\boldsymbol{r}}_x, \boldsymbol{u} \rangle}$	$g^{-\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle \frac{\mathrm{J}(\mathsf{gid})}{\mathrm{F}(\mathsf{gid})}} \cdot D^{d_1^* \widetilde{r}} g_q^{d_2^* \widehat{r}_q \widetilde{r}}$	$(D^{\widetilde{r}} g_q^{\widehat{r}_q \widetilde{r}}) \cdot g^{-\frac{\langle \boldsymbol{\psi}_x, \boldsymbol{u} \rangle}{\mathrm{F}(\mathsf{gid})}}$

Table 43: Simulated secret keys used in upper identity-hiding game

$\mathsf{F}(\mathsf{gid}) \neq 0 \mod N$ for any identity (since we assume $N > sk_1'$ for any reasonable value of N, s and k_1'). To prove this, we need the following Lemma 16.

There have some restrictions over the key queries to the key generation oracle i.e.,

- Adversary $\mathscr{A}$ can not query for the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ such that $i = i^* \wedge \mathsf{id}_{\ell^*} \neq b^* \wedge \mathsf{gid} = \mathsf{gid}^*$.
- In Case 1, the adversary can make a query for the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ where $i = i^* \wedge \mathsf{id}_{\widetilde{\ell}} = \widetilde{b} \wedge \mathsf{gid} = \mathsf{gid}^*$. If the adversary makes a key query for the challenge index position i^* with $\mathsf{id}_{\widetilde{\ell}} \neq \widetilde{b}$ and $\mathsf{gid} = \mathsf{gid}^*$, then the challenger $\mathscr{B}$ aborts.

Ciphertext simulation. To generate the challenge ciphertext, the challenger chooses the exponents as follows.

$$\begin{aligned} \forall j \in [m],\ \ \sigma_j, \nu_j, \phi_j &\leftarrow \mathbb{Z}_N,\ \ \tau, t_q \leftarrow \mathbb{Z}_N, \\ \forall x \in [\widehat{n}],\ \ e_x, f_x, d_x, \widetilde{s}_x &\leftarrow \mathbb{Z}_N \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\},\ \ \widetilde{w}_{y,\ell,b}, \upsilon_{y,\ell,b} &\leftarrow \mathbb{Z}_N \end{aligned}$$

For the challenge group identity gid^*, $\mathscr{B}$ can compute as $\mathrm{H}(\mathsf{gid}^*) = \vartheta' \prod_{i \in \mathcal{V}^*} \vartheta_i = g_p^{d_p^*} g_q^{d_q^*}$ and $\mathrm{H}_q(\mathsf{gid}^*)^\beta = \vartheta_q'^{\beta} \prod_{i \in \mathcal{V}^*} \vartheta_{q,i}^{\beta} = g_q^{d_q^*}$, where $d_q^*, d_p^* \in \mathbb{Z}_N$. Now, the challenger generates row and columns components as the tables below.

Table 44: Row components computation for $x \in [\widehat{n}], j \in [m]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{\widetilde{s}_x}$	$F_{q,x,j}^{\widetilde{s}_x \tau}$	$E_q^{\widetilde{s}_x t_q}$	$\mathrm{H}_q(\mathsf{gid}^*)^{\beta \widetilde{s}_x t_q}$	$e(g_q, g_q)^{\nu_j} \cdot G_{q,x,j}^{\widetilde{s}_x t_q} \cdot e(f_q, Y_{q,x,j})^{\widetilde{s}_x t_q}$
$x = x^*$	$(Dg_q^{\widehat{r}_q})^{\widetilde{s}_x \widetilde{r}_{x,j}}$	$(Fh_q^{\widehat{r}_q})^{\widetilde{s}_x \widetilde{r}_{x,j} \tau}$	$(Ag_q^{t_q})^{\widetilde{s}_x}$	$A^{d_p^* \widetilde{s}_x} g_q^{d_q^* \widetilde{s}_x t_q}$	$e(g,g)^{\nu_j} e(g, Ag_q^{t_q})^{\widetilde{s}_x \alpha_{x,j}} \cdot e(B, A)^{d_1^2 \psi_{x,j} \widetilde{s}_x} e(f_q, g_q)^{\psi_{x,j} \widetilde{s}_x t_q}$
$x < x^*$	$g^{\widetilde{s}_x \sigma_j}$	$h^{\widetilde{s}_x \tau \nu_j}$	g^{e_x}	$\mathrm{H}(\mathsf{gid}^*)^{d_x}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Table 45: Column components computation for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge \ell > \widetilde{\ell}) \vee (y = y^* \wedge \ell \notin [\widetilde{\ell}] \cup \{\ell^*\}) \vee (y = y^* \wedge \ell = \widetilde{\ell} \wedge b > \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\widetilde{w}_{y,\ell,b} \tau}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} g^{\widetilde{w}_{y,\ell,b}}$
$y = y^* \wedge \ell = \widetilde{\ell} \wedge b = \widetilde{b}$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} \cdot h^{\tau \widetilde{w}_{y,\ell,b}} \cdot T^{-1}$	$A^{-\widetilde{c}_{y,\ell,b}/\tau} g^{\widetilde{w}_{y,\ell,b}}$
$(y < y^*) \vee (y = y^* \wedge \ell < \widetilde{\ell}) \vee (y = y^* \wedge \ell \in ([\widetilde{\ell} - 1] \cup \{\ell^*\}) \wedge b < \widetilde{b})$	$g_q^{\widetilde{c}_{y,\ell,b} t_q} h^{\tau \widetilde{w}_{y,\ell,b}} g_p^{\nu_{y,\ell,b}}$	$g^{\widetilde{w}_{y,\ell,b}}$

After seeing the challenge ciphertext the adversary guess a bit $\mathfrak{b}'$ and it forwarded to a modified-1 D3DH challenger of assumption 1.

Analysis of simulation. If $T = g_p^{abc}$ then $\mathscr{B}$ simulates the view of sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell}+\widetilde{b}-1,\widetilde{b}+1 \mod 2}$ otherwise if T is a random group element from the subgroup $\mathbb{G}_p$ then $\mathscr{B}$ simulates the view of sub-hybrid $\mathsf{Hyb}_{3,\widetilde{\ell},\widetilde{b}}$. Therefore, if the adversary $\mathscr{A}$ wins the game with an advantage $\epsilon(\cdot)$

then $\mathscr{B}$ breaks the assumption 1 with same $\epsilon(\cdot)$ advantage.

Case 2. (otherwise) The proof technique is similar to the proof of Claim 8 and Lemma 13.

In this case, we use the same proof strategy as used in Claim 8 and Lemma 13, where the reduction algorithm does not need to know the value of the group element $g^{\widehat{r}\langle \boldsymbol{r}_{x^*}, \boldsymbol{u}\rangle c_{y^* m\widetilde{\ell},\widetilde{b}}}$ for answering the key queries.

$\underline{\mathbf{Hyb}_3 \approx \mathbf{Hyb}_4}$: The proof of the above indistinguishability of hybrids Hyb_3 and Hyb_4 are identical. No adversary can not distinguish between these hybrids with non-negligible advantage.

$\underline{\mathbf{Hyb}_4 \approx \mathbf{Hyb}_5}$: The indistinguishable of the hybrids Hyb_4 and Hyb_5 can be classified into two cases.

Case 1. If $y^* \neq \widetilde{n}$, then both the hybrids Hyb_4 and Hyb_5 are identical.

Case 2. For $y^* = \widetilde{n}$ then the indistinguishability follows from the sequence of hybrid games which is similar to the claim 9. In particular, Hyb_4 as described above is similar to the hybrid 2 and Hyb_5 is identical with the hybrid 7 as described in the claim 9. Thus this indistinguishable follows from the claim 9.

This concludes the upper identity-hiding security game.

Lemma 15 *If the joint-D3DH assumption 7 holds over the bilinear group $\mathbb{BG}$, then our EIPL-IBIPFE satisfies the message-hiding security as per Definition 17.*

Proof. Suppose, $\mathscr{A}$ is a PPT adversary against the message-hiding security of the our EIPL-IBIPFE scheme. We construct an algorithm $\mathscr{B}$ for breaking the joint-D3DH assumption that uses $\mathscr{A}$ as a subroutine. To prove the message-hiding security, we consider two hybrid games. The first game is same as the original game message-hiding as in definition 17. In the next game, we change the distribution of the master public key, secret key and the challenge ciphertext, where we first sample a random vector $\widetilde{\boldsymbol{\psi}}_x$ and set $\boldsymbol{\psi}_{\boldsymbol{x}} = \mathbf{F}^\top \widetilde{\boldsymbol{\psi}}_x$ for some $x \in [\widehat{n}]$. The matrix $\mathbf{F}$ is a full rank matrix chosen such that $\mathbf{F}(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)}) = \boldsymbol{e}_1$ where $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ are the challenge message vectors submitted by the $\mathscr{A}$ and $\boldsymbol{e}_1$ denotes a m length vector as $(1,0,\ldots,0)^\top$. Assuming the joint-D3DH assumption 7 holds over the bilinear group $\mathbb{BG}$, we show that the adversary can distinguish between the challenge ciphertexts with negligible probability.

Game 0: The game is same as the message-hiding as per Definition 17.

Game 2: The game is exactly same as the previous game except for each identity gid, the challenger samples the master secret key msk as follows:

(a) Samples uniformly random vector $\widetilde{\boldsymbol{\psi}}_x = (\widetilde{\psi}_{x,1}, \widetilde{\psi}_{x,2}, \cdots, \widetilde{\psi}_{x,m})$ for all $\widetilde{\psi}_{x,j} \in \mathbb{Z}_N$ and $x \in [\widehat{n}]$.
(b) Samples a uniformly chosen full rank matrix $\mathbf{F} \in \mathbb{Z}_N^{m\times m}$ satisfying the relation $\mathbf{F}(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)}) = \boldsymbol{e}_1$ where $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ are the challenge messages vectors of length m.
(c) Sets $\boldsymbol{\psi}_x = \mathbf{F}^\top \widetilde{\boldsymbol{\psi}}_x$ instead of sampling uniformly random as in Game 0.

In the adversary's view, the master public key mpk, secret key associated with the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ where $i \geq i^*$ and the challenge ciphertext ct is simulated as below.

Public key: All the components of mpk are generated similarly as Game 0 except $Y_{q,x,j} = (g_q^{\beta \mathbf{F}\boldsymbol{\psi}_x})_j$ and $Y_{x,j} = (g^{\mathbf{F}\boldsymbol{\psi}_x})_j$ where $(\boldsymbol{g})_j$ represents the j-th group elements from the vector $\boldsymbol{g} = (g_1, g_2, \ldots, g_m)$.

Secret key: We consider $\mathcal{V} \subseteq \{1,\ldots,k'\}$ be the set of all i for which the i-th component of queried group identity is non-zero i.e., $\mathsf{gid}_i = 1$. The secret key $\mathsf{sk}_{\boldsymbol{u}}$ components corresponding to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ is set as

$$K_1 = g^{\langle \widetilde{\boldsymbol{\alpha}}_x, \boldsymbol{u}\rangle} (\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell})^{\widehat{r}\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle}, \;\; K_2 = f^{\langle \widetilde{\boldsymbol{\psi}}_x, \mathbf{F}\boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid})^r, \;\; K_3 = g^r$$

Challenge ciphertext: For the challenge group identity gid^*, consider $\mathcal{V}^* \subseteq \{1,\ldots,k'\}$ be the set of all i such that $\mathsf{gid}^*_i = 1$. All the components of the ciphertext except $A_x, B_x, I_{x,j}$ for $x > x^*$ and $x = x^*$ are similarly generated as the Game 0.

Table 46: Computing row components of the ciphertext for $x \in [\widehat{n}]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$g_q^{\beta t s_x}$	$\mathrm{H}_q(\mathsf{gid}^*)^{t s_x \beta}$	$\left(e(g_q, g_q)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)}))} \cdot e(g_q, f_q)^{\beta t s_x \mathbf{F}^\top \widetilde{\boldsymbol{\psi}}_x}\right)_j$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	$g^{t s_x}$	$\mathrm{H}(\mathsf{gid}^*)^{t s_x}$	$\left(e(g,g)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)}))} \cdot e(g,f)^{t s_x \mathbf{F}^\top \widetilde{\boldsymbol{\psi}}_x}\right)_j$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	$C_1^{s_x \mathrm{J}(\mathsf{gid}^*)}$	$e(g,g)^{f_x \phi_j} \cdot e(f,f)^{f_x \phi_j}$

Since, $\mathbf{F} \in \mathbb{Z}_N^{m \times m}$ is an orthogonal matrix, then the following two distributions are equivalent.

$$\{\boldsymbol{\psi}_x : \boldsymbol{\psi}_x \leftarrow \mathbb{Z}_N^m, x \in [\widehat{n}]\} \equiv \{\mathbf{F}^\top \widetilde{\boldsymbol{\psi}}_x : \widetilde{\boldsymbol{\psi}}_x \leftarrow \mathbb{Z}_N^m, x \in [\widehat{n}]\}$$

Therefore, the advantage of any PPT adversary $\mathscr{A}$ can distinguish between the Game 0 and Game 1 with negligible probability.

Public key simulation: Without loss of generality, we assume that the adversary makes the maximum number of Q queries and the challenge group identity gid^* and challenge index i^*. Now, the simulator chooses an integer $k_1' \leftarrow [k']$, sets an integer $s = 10Q$, a random k'-length vector $\boldsymbol{z} = (z_i) \leftarrow \mathbb{Z}_s^{k'}$ and a value $z' \leftarrow \mathbb{Z}_s$. Additionally, the simulator also chooses a random value $w' \leftarrow \mathbb{Z}_N$ and a uniformly random k'-length vector $\boldsymbol{w} = (w_i) \leftarrow \mathbb{Z}_N^{k'}$. All these values are kept secret to the simulator.

Let us consider $\mathcal{V}^* \subseteq \{1,2,\ldots,k'\}$ be the set of all i for which the challenge identity $\mathsf{gid}^*_i = 1$. Let $\mathcal{V}^* = \{i_1, i_2, \ldots, i_\kappa\}$. Now, we choose the z_i values from $\boldsymbol{z}$ which correspond to the index set $\mathcal{V}^*$ and then set $\sum_{i \in \mathcal{V}^*} z_i = k_1' s - z'$ for uniformly chosen $k_1' \in [k']$. Now, we define the function $\mathrm{K}(\mathsf{gid})$ as

$$\mathrm{K}(\mathsf{gid}) = \begin{cases} 0, & \text{if } z' + \sum_{i \in \mathcal{V}} z_i \equiv 0 \mod s \\ 1, & \text{elsewhere} \end{cases}$$

So, from the above definition of the function K, we can say that $\mathrm{K}(\mathsf{gid}^*) = 0$ and for all $\mathsf{gid}(\neq \mathsf{gid}^*)$ it becomes non-zero. Additionally, we set two functions as $\mathrm{F}(\mathsf{gid}) = N - s k_1' + z' + \sum_{i \in \mathcal{V}} z_i$ and $\mathrm{J}(\mathsf{gid}) = w' + \sum_{i \in \mathcal{V}} w_i$. The simulator assigns the public parameters $\vartheta' = f^{N - k_1' s + z'} \cdot g^{w'} = g_p^{d_p'} g_q^{d_q'}$ and $\vartheta_i = f^{z_i} g^{w_i} = g_p^{d_{p,i}} g_q^{d_{q,i}}$, From the adversarial perspective, the distribution of the public parameters is identical to the real construction.

Suppose on the contrary, there exist a PPT adversary $\mathscr{A}$ that can distinguish between the Game 0 and Game 1 with non-negligible advantage $\epsilon(\cdot)$. Then, we construct a PPT adversary $\mathscr{B}$ that breaks the joint-D3DH assumption 7 with the same non-negligible advantage. Let us consider, $\mathscr{B}$ gets an instance

$$(\mathbb{BG}, A_p = g_p^a, D_p = g_p^d, C_p = g_p^c, A_q = g_q^a, D_q = g_q^d, C_q = g_q^c, S_p = g_p^\tau, S_q = g_q^\tau)$$

of the joint-D3DH assumption 7, where $\mathbb{BG} = (p, q, N, \mathbb{G}, \mathbb{G}_T, g, e(\cdot,\cdot))$ is a group description. The elements $a, d, c, \tau \leftarrow \mathbb{Z}_N$ are random integers and the elements g_p^τ, g_q^τ are either g_p^{adc}, g_q^{adc} or random group elements from the subgroups $\mathbb{G}_p$, $\mathbb{G}_q$ respectively. In the following, we discuss the simulation of master public key mpk. Before going to the simulation, $\mathscr{B}$ computes

$$A_p A_q = g_p^a g_q^a = g^a = A;\ D_p D_q = g_p^d g_q^d = g^d = D;\ C_p C_q = g_p^c g_q^c = g^c = C$$

Therefore, the challenge component is either $S = S_p S_q = g^\tau$ or a random element from the group $\mathbb{G}$. The algorithm $\mathscr{B}$ works as follows:

The adversary $\mathscr{B}$ implicitly sets the following vector of length m as

$$\boldsymbol{a}_x = (a, a_{x,2}, \ldots, a_{x,m});\ \boldsymbol{d} = (d, \ldots, d);\ \boldsymbol{c} = (c, \ldots, c)$$

where it random samples $a_{x,2}, \ldots, a_{x,m} \leftarrow \mathbb{Z}_N$. We define the notion $\boldsymbol{u} \odot \boldsymbol{v}$ by component wise multiplication of the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. In this case, $\boldsymbol{a} \odot \boldsymbol{d} = (ad, a_{x,2}d, \ldots, a_{x,m}d) = d\boldsymbol{a}$. To generate the public key, we implicitly set $\widetilde{\boldsymbol{\psi}}_x = \boldsymbol{a}_x$ and randomly chooses some exponents as follows:

$$\begin{aligned} \forall \ell \in [k], b \in \{0,1\}, &\quad \delta_{\ell,b} \leftarrow \mathbb{Z}_N, \gamma_{\ell,b} \leftarrow \mathbb{Z}_p \\ \forall y \in [\widetilde{n}], \ell \in [k], b \in \{0,1\} &\quad c_{y,\ell,b} \leftarrow \mathbb{Z}_N \\ \forall j \in [m], x \in [\widehat{n}] &\quad r_{x,j}, \alpha_{x,j} \leftarrow \mathbb{Z}_N \end{aligned}$$

Additionally, consider g_p, h_p, f_p are generators of $\mathbb{G}_p$ and g_q, h_q, f_q are generators of the group $\mathbb{G}_q$ and randomly chose $\beta, \widehat{r} \leftarrow \mathbb{Z}_N$. Then, it sets the master public key as

$$\mathsf{mpk} = \left(\begin{array}{c} \mathbb{BG}, h, g, f = g^d = D, f_q = g_q^d = D_q, \\ \vartheta', \vartheta_q^{'\beta}, \boldsymbol{\vartheta}, \{\vartheta_{q,i}^\beta\}, \mathrm{H}, \mathrm{H}_q, E_q = g_q^\beta \\ \left\{ \begin{array}{c} E_{x,j} = g^{\widehat{r} r_{x,j}}, F_{x,j} = h^{\widehat{r} r_{x,j}}, G_{x,j} = e(g,g)^{\alpha_{x,j}} \\ E_{q,x,j} = g_q^{\beta \widehat{r} r_{x,j}}, F_{q,x,j} = h_q^{\beta \widehat{r} r_{x,j}}, G_{q,x,j} = e(g_q, g_q)^{\beta \alpha_{x,j}} \\ Y_{x,j} = (g^{\mathbf{F}^\top \boldsymbol{a}_x})_j, Y_{q,x,j} = (g_q^{\beta \mathbf{F}^\top \boldsymbol{a}_x})_j, \end{array} \right\}_{x \in [\widehat{n}], j \in [m]} \\ \{H_{y,\ell,b} = g^{c_{y,\ell,b}}\}_{(y,\ell,b) \in [\widetilde{n}] \times [k] \times \{0,1\}}, \\ \left\{ \widetilde{V}_{\ell,b} = g^{\delta_{\ell,b}} g_p^{\gamma_{\ell,b}}, V_{\ell,b} = h^{\delta_{\ell,b}} \right\}_{(\ell,b) \in [k] \times \{0,1\}} \end{array} \right)$$

Here, the exponent $g^{\boldsymbol{a}_x}, g_q^{\boldsymbol{a}_x}$ are computed as

$$g^{\boldsymbol{a}_x} = (g^a, g^{a_{x,2}}, \ldots, g^{a_{x,m}}) = g^{\widetilde{\boldsymbol{\psi}}_x}; g_q^{\boldsymbol{a}_x} = (g_q^a, g_q^{a_{x,2}}, \ldots, g_q^{a_{x,m}}) = g_q^{\widetilde{\boldsymbol{\psi}}_x}$$

Hence, the public key components are properly generated by using the joint-D3DH instances as described above.

Secret key simulation: $\mathscr{B}$ answers the secret key $\mathsf{sk}_{\boldsymbol{u}}$ associated to the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ as describe below. We consider two cases. Before going further, we consider $\mathcal{V} \subseteq \{1, \ldots, k'\}$ be the set of all j such that $\mathsf{gid}_j = 1$. Let $i = (x, y)$ be the row wise representation with $i \geq i^*$.

Case 1. If the group identity $\mathsf{gid} \neq \mathsf{gid}^*$, $\mathscr{B}$ simulates the secret key as follows:

$$K_1 = g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u} \rangle} \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell} \right)^{\widehat{r} \langle \boldsymbol{r}_x, \boldsymbol{u} \rangle}$$

$$
\begin{aligned}
K_2 &= g^{-\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle \frac{\mathrm{J(gid)}}{\mathrm{F(gid)}}} \mathrm{H(gid)}^r \\
&= f^{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle} \left(f^{\mathrm{F(gid)}} g^{\mathrm{J(gid)}}\right)^{-\frac{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle}{\mathrm{F(gid)}}} \left(f^{\mathrm{F(gid)}} g^{\mathrm{J(gid)}}\right)^r \\
&= f^{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle} \left(f^{\mathrm{F(gid)}} g^{\mathrm{J(gid)}}\right)^{r-\frac{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle}{\mathrm{F(gid)}}} \\
&= f^{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle} \left(f^{\mathrm{F(gid)}} g^{\mathrm{J(gid)}}\right)^{r'} \\
&= f^{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle} \mathrm{H(gid)}^{r'} = f^{\langle \tilde{\boldsymbol{\psi}}_x, \mathbf{F}\boldsymbol{u}\rangle} \mathrm{H(gid)}^{r'} \\
K_3 &= g^r g^{-\langle (a, a_{x,2}, \ldots, a_{x,m}), \mathbf{F}\boldsymbol{u}\rangle \cdot \frac{1}{\mathrm{F(gid)}}} \\
&= g^{r-\frac{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle}{\mathrm{F(gid)}}} = g^{r'}
\end{aligned}
$$

We implicitly set $r' = r - \frac{\langle \tilde{\boldsymbol{\psi}}_x, \mathbf{F}\boldsymbol{u}\rangle}{\mathrm{F(gid)}}$. So from the construction of K function, it can conclude that $\mathrm{K(gid)} \neq 0$ for any key query corresponding to the group identity gid. This implies that the function $\mathrm{F(gid)} \neq 0 \mod N$ for any identities (as we assume $N > sk_1'$ for any reasonable value of N, s and k_1'). To prove this, we need the following Lemma 16.

Case 2. If $\mathsf{gid} = \mathsf{gid}^*$, $\mathscr{B}$ responds the secret keys as follows:

$$
\begin{aligned}
K_1 &= g^{\langle \boldsymbol{\alpha}_x, \boldsymbol{u}\rangle} \left(\prod_{\ell \in [k]} H_{y,\ell,\mathsf{id}_\ell}\right)^{\hat{r}\langle \boldsymbol{r}_x, \boldsymbol{u}\rangle} \\
K_2 &= g^{d\mu} \left(f^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r\right) \\
&= g^{\langle \boldsymbol{a}_x \odot \boldsymbol{d}, \mathbf{F}\boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r\right) \\
&= f^{\langle (a, a_{x,2}, \ldots, a_{x,m}), \mathbf{F}\boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r\right) \\
&= f^{\langle \boldsymbol{a}_x, \mathbf{F}\boldsymbol{u}\rangle} \left(f^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r\right) \\
&= f^{\langle \tilde{\boldsymbol{\psi}}_x, \mathbf{F}\boldsymbol{u}\rangle} \mathrm{H}(\mathsf{gid}^*)^r \\
K_3 &= g^r
\end{aligned}
$$

where the second equality follows from the fact that $\langle \boldsymbol{a}_x \odot \boldsymbol{d}, \mathbf{F}\boldsymbol{u}\rangle = d\mu$ with $\mu \in \mathbb{Z}_N$ is known to the challenger $\mathscr{B}$. So from the formation of the matrix $\mathbf{F}$, we have $\mathbf{F}(\boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)}) = \boldsymbol{e}_1$ and for $\mathsf{gid} = \mathsf{gid}^*$, the queried secret key associated with the vector $\boldsymbol{u}$ satisfies the condition $\langle \boldsymbol{v}^{(0)} - \boldsymbol{v}^{(1)}, \boldsymbol{u}\rangle = 0$. Therefore, we have $\langle \boldsymbol{e}_1, \mathbf{F}\boldsymbol{u}\rangle = 0$ which implies that the inner product $\langle \boldsymbol{a}_x \odot \boldsymbol{d}, \mathbf{F}\boldsymbol{u}\rangle = \langle (ad, a_{x,2}d, a_{x,3}d, \ldots, a_{x,m}d), \mathbf{F}\boldsymbol{u}\rangle = d\mu$ for some $\mu \in \mathbb{Z}_N$.

Challenge ciphertext simulation: To the generates the challenge ciphertext, $\mathscr{B}$ chooses the random exponent $t \leftarrow \mathbb{Z}_N$ and implicitly set $t = c$. In the following, we show that how $\mathscr{B}$ simulates the challenges ciphertext using the joint-D3DH instances.
For $x \geq x^*$, $A_x = C_q^{\beta s_x}$

$$
\begin{aligned}
B_x &= C_q^{\beta s_x \mathrm{J}_q(\mathsf{gid}^*)} \\
&= \mathrm{H}_q(\mathsf{gid}^*)^{\beta s_x t} \quad [\text{Since, } N - k_1' s + z' + \sum_{i \in \mathcal{V}^*} z_i = N = pq \implies f_q^N = 1] \\
I_{x,j} = I_{x,j}^1 &= \left(e(g_q, g_q)^{\mathbf{F}^\top (\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g_q, g_q)^{\beta s_x c \boldsymbol{\alpha}_x} e(g_q, g_q^{cd\mathbf{F}^\top (a, a_{x,2}, \ldots, a_{x,m})})^{\beta s_x}\right)_j \\
&= \left(e(g_q, g_q)^{\mathbf{F}^\top (\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g_q, g_q)^{\beta s_x c \boldsymbol{\alpha}_x} e(g_q, g_q)^{\beta s_x cd\mathbf{F}^\top (a, a_{x,2}, \ldots, a_{x,m})}\right)_j
\end{aligned}
$$

$$= \left(e(g_q, g_q)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g_q, g_q)^{\beta s_x c \boldsymbol{\alpha}_x} e(g_q, f_q)^{\beta s_x c \mathbf{F}^\top(a, a_{x,2}, \ldots, a_{x,m})} \right)_j$$
$$= \left(e(g_q, g_q)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g_q, g_q)^{\beta s_x t \boldsymbol{\alpha}_x} e(g_q, f_q)^{\beta s_x t \mathbf{F}^\top \tilde{\boldsymbol{\psi}}_x} \right)_j$$

For $x = x^*$, $A_x = C^{s_x}$, $B_x = C^{s_x \mathsf{J}(\mathsf{gid}^*)}$,

$$I_{x,j} = I^2_{x,j} = \left(e(g,g)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g,g)^{s_x c \boldsymbol{\alpha}_x} e(g, g^{cd\mathbf{F}^\top(a, a_{x,2}, \ldots, a_{x,m})})^{s_x} \right)_j$$
$$= \left(e(g,g)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g,g)^{s_x c \boldsymbol{\alpha}_x} e(g,g)^{s_x cd\mathbf{F}^\top(a, a_{x,2}, \ldots, a_{x,m})} \right)_j$$
$$= \left(e(g,g)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g,g)^{s_x c \boldsymbol{\alpha}_x} e(g,f)^{s_x c \mathbf{F}^\top(a, a_{x,2}, \ldots, a_{x,m})} \right)_j$$
$$= \left(e(g,g)^{\mathbf{F}^\top(\mathbf{F}\boldsymbol{v}^{(0)} - \mathfrak{b}\boldsymbol{e}_1)} e(g,g)^{s_x t \boldsymbol{\alpha}_x} e(g,f)^{s_x t \mathbf{F}^\top \tilde{\boldsymbol{\psi}}_x} \right)_j$$

Table 47: Simulation of row components of the challenge ciphertext for $x \in [\widehat{n}]$

	$R_{x,j}$	$\widetilde{R}_{x,j}$	A_x	B_x	$I_{x,j}$
$x > x^*$	$E_{q,x,j}^{s_x}$	$F_{q,x,j}^{s_x \tau}$	$C_q^{\beta s_x}$	$C_q^{\beta s_x \mathsf{J}(\mathsf{gid}^*)}$	$I_{x,j}^1$
$x = x^*$	$E_{x,j}^{s_x}$	$F_{x,j}^{s_x \tau}$	C^{s_x}	$C^{s_x \mathsf{J}(\mathsf{gid}^*)}$	$I_{x,j}^2$
$x < x^*$	$g^{s_x \sigma_j}$	$h^{s_x \tau \nu_j}$	g^{e_x}	$C^{s_x \mathsf{J}(\mathsf{gid}^*)}$	$e(g,g)^{f_x \phi_j} \cdot e(D,D)^{f_x \phi_j}$

Table 48: Computing column components of the ciphertext for $(y, \ell, b) \in [\widetilde{n}] \times [k] \times \{0,1\}$

	$C_{y,\ell,b}$	$\widetilde{C}_{y,\ell,b}$
$(y > y^*) \vee (y = y^* \wedge (\ell, b) \neq (\ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}}$
$(y < y^*) \vee (y, \ell, b) = (y^*, \ell^*, b^*))$	$H_{y,\ell,b}^t \cdot h^{w_{y,\ell,b}\tau} V_{\ell,b}^{v_{y,\ell,b}\tau}$	$g^{w_{y,\ell,b}} \cdot \widetilde{V}_{\ell,b}^{v_{y,\ell,b}}$

Guess. If $\mathscr{A}$ guesses the challenge bit $\mathfrak{b} \leftarrow \{0,1\}$ correctly, then $\mathscr{B}$ returns 1 otherwise it outputs 0. We consider $\boldsymbol{w}_x = dc(a, a_{x,2}, \ldots, a_{x,m}) = (\tau, dca_{x,2}, \ldots, dca_{x,m})$ where g^τ and g_q^τ are the challenge elements. If $\tau = adc$, then all the secret key and the challenge ciphertext is properly distributed. In particular, the challenge ciphertext is an encryption of message vector $\boldsymbol{v}^{(\mathfrak{b})}$. Therefore, in this case, $\mathscr{A}$ outputs $\mathfrak{b} = \mathfrak{b}'$ with advantages $\frac{1}{2} + \mathsf{negl}(\lambda)$ where $\mathsf{negl}(\lambda)$ is the advantage of $\mathscr{A}$ in the message-hiding security game of the EIPL-IBIPFE. Otherwise, if τ is randomly generated from $\mathbb{Z}_N$ then the challenge ciphertext components $I_{x,j}$ uniform element from the target group $\mathbb{G}_T$. So the $\mathscr{A}$ can not get any information about the challenge bit $\mathfrak{b}$ from this component. So, $\mathscr{A}$ wins the game with the probability $\frac{1}{2}$. Hence, from the hardness assumption 7, it can conclude that $\mathscr{A}$ has a non-negligible advantage against the proposed EIPL-IBIPFE scheme achieves the selective security. This completes the message-hiding security. □

Therefore, it concludes the security of our EIPL-IBIPFE. □

Lemma 16 *For any Q-secret key query corresponding to the identities* $\mathsf{gid}_1, \mathsf{gid}_2, \ldots, \mathsf{gid}_Q$ *to the key generation oracle, the probabilities of* $\mathrm{K}(\mathsf{gid}^{(\ell)}) = 1$ *with* $z' + \sum_{i \in \mathcal{V}^*} z_i = k_1'$*s is non-negligible for all* ℓ.

Proof. For any set of q-queries corresponding to the identities $\mathsf{gid}_1, \mathsf{gid}_2, \ldots, \mathsf{gid}_Q$ and the challenge identity gid^*, we compute the following probability.

$$\begin{aligned}
&\Pr\left[\bigwedge_{\ell=1}^{Q}(\mathrm{K}(\mathsf{gid}^{(\ell)})=1) | (z' + \sum_{i\in\mathcal{V}^*} z_i) = k_1' s\right] \\
&= \Pr\left[\bigwedge_{\ell=1}^{Q}(\mathrm{K}(\mathsf{gid}^{(\ell)})=1 | \mathrm{K}(\mathsf{gid}^*)=0)\right] \\
&= \left(1 - \Pr\left[\bigvee_{\ell=1}^{Q}(\mathrm{K}(\mathsf{gid}^{(\ell)})=0) | \mathrm{K}(\mathsf{gid}^*)=0)\right]\right) \\
&\geq \left(1 - \sum_{\ell=1}^{Q}\Pr\left[(\mathrm{K}(\mathsf{id}^{(\ell)})=0) | \mathrm{K}(\mathsf{gid}^*)=0)\right]\right) \\
&= \left(1 - \frac{Q}{s}\right) = 0.9
\end{aligned}$$

We can optimize the last equation by setting $s = 10Q$ (as we did in the simulation), where Q is the maximum number of queries. The above result shows that for all queried identities in the key generation oracle except the challenge identity gid^*, the K values should be equal to 1 with overwhelming probability. □

8 EIPL-IBIPFE from ABIPFE and MFE

Let us consider ABIPFE = (ABIPFE.Setup, ABIPFE.KeyGen, ABIPFE.Enc, ABIPFE.Dec) be an attribute-based IPFE scheme for a class of functions $\mathscr{F}_\lambda$, a predicate space $\mathscr{X}_\lambda$ and a message space $\mathscr{Y}_\lambda$ and MFE = (MFE.Setup, MFE.PK-Enc, MFE.SK-Enc, MFE.KeyGen, MFE.Dec) be a mixed FE for a class of functions $\mathscr{F}_\lambda$. We provide a generic transformation of EIPL-IBIPFE from ABIPFE and MFE. Let $\bar{k}$ be the functionality index representing the function class $\{f_{i,\ell,b,\mathsf{gid}}\}$ defined in the special encryption algorithm below.

Setup($1^\lambda, n, 1^k, 1^{k'}, 1^m$) → (msk, mpk, key): On input integers λ, k, k', m, n as unary, the setup algorithm performs the following steps:

- runs the setup algorithm of ABIPFE and computes its master public and master secret key pair as $(\mathsf{ABIPFE.msk}, \mathsf{ABIPFE.mpk}) \leftarrow \mathsf{ABIPFE.Setup}(1^\lambda, 1^m, 1^{\kappa+k'})$ where $\kappa = \kappa(\lambda)$ is the length of an MFE ciphertext.
- runs the setup algorithm of MFE and computes $(\mathsf{MFE.msk}, \mathsf{MFE.mpk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda, 1^{\bar{k}})$.
- outputs mpk = (ABIPFE.mpk, MFE.mpk), msk = (ABIPFE.msk, MFE.msk), key = msk.

KeyGen(msk, i, id, gid, $\boldsymbol{u}$) → $\mathsf{sk}_{\boldsymbol{u}}$: The key generation algorithm takes input the master secret key msk = (ABIPFE.msk, MFE.msk), an index $i \in [n]$, an user identity $\mathsf{id} \in \{0,1\}^k$, a group identity $\mathsf{gid} \in \{0,1\}^{k'}$, and a vector $\boldsymbol{u} \in \mathbb{Z}^m$. It performs as follows:

- computes $\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{MFE.msk}, (i, \mathsf{id}, \mathsf{gid}))$.
- define a circuit $\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot)$ as

$$\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{MFE.ct}, \mathsf{gid}') = \begin{cases} 1 - \mathsf{MFE.Dec}(\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}}, \mathsf{MFE.ct}) & \text{if } \mathsf{gid} = \mathsf{gid}' \\ \perp & \text{otherwise.} \end{cases}$$

- computes $\mathsf{ABIPFE.sk}_{\boldsymbol{u}} \leftarrow \mathsf{ABIPFE.KeyGen}(\mathsf{ABIPFE.msk}, \mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot), \boldsymbol{u})$.
- returns $\mathsf{sk}_{\boldsymbol{u}} = \mathsf{ABIPFE.sk}_{\boldsymbol{u}}$.

Enc(mpk, gid′, $\boldsymbol{v}$) → $\mathbf{ct}_{\boldsymbol{v}}$: The encryption algorithm takes input the master public key mpk = (ABIPFE.mpk, MFE.mpk), a message vector $\boldsymbol{v} \in \mathbb{Z}^m$ and a group identity gid′. It performs the following mechanisms:

- computes MFE.ct ← MFE.PK-Enc(MFE.mpk).
- computes $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}'), \boldsymbol{v})$.
- returns $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$.

SplEnc(key, gid′, $\boldsymbol{v}$, (i^*, ℓ^*, b^*)) → $\mathbf{ct}_{\boldsymbol{v}}$: On input key = msk, message vector $\boldsymbol{v} \in \mathbb{Z}^m$, a group identity gid′, a index-position-bit tuple (i^*, ℓ^*, b^*), the special encryption algorithm performs the following steps.

- construct a function $f_{i^*,\ell^*,b^*,\mathsf{gid}'}(j, \mathsf{id}, \mathsf{gid})$ as follows:

$$f_{i^*,\ell^*,b^*,\mathsf{gid}'}(j, \mathsf{id}, \mathsf{gid}) = \begin{cases} 1, & \text{if } (j \geq i^* + 1) \vee (i^*, \ell^*) = (j, \perp) \vee \\ & (i^*, \mathsf{id}_{\ell^*}) = (j, 1 - b^*) \wedge (\mathsf{gid} = \mathsf{gid}') \\ 0, & \text{otherwise} \end{cases}$$

- computes $\mathsf{MFE.ct} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{MFE.msk}, f_{i^*,\ell^*,b^*,\mathsf{gid}'})$.
- computes $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}'), \boldsymbol{v})$.
- returns $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$

Dec($\mathbf{sk}_{\boldsymbol{u}}$, $\mathbf{ct}_{\boldsymbol{v}}$) → $\boldsymbol{\zeta}$/ ⊥: On input the secret key $\mathsf{sk}_{\boldsymbol{u}}$ and ciphertext $\mathsf{ct}_{\boldsymbol{v}}$, the decryptor runs the ABIPFE and returns the output $\mathsf{ABIPFE.Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}})$.

8.1 Correctness.

For any $\lambda, n \in \mathbb{N}$, a message vector $\boldsymbol{v} \in \mathscr{Y}_\lambda$, the master public key, master secret key pair $(\mathsf{ABIPFE.msk}, \mathsf{ABIPFE.mpk}) \leftarrow \mathsf{ABIPFE.Setup}(1^\lambda, 1^m, 1^{\kappa+k'})$ and $(\mathsf{MFE.msk}, \mathsf{MFE.mpk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda, 1^{\bar{k}})$, the secret keys are generated from ABIPFE keys as $\mathsf{sk}_{\boldsymbol{u}} = \mathsf{ABIPFE.sk}_{\boldsymbol{u}}$ for $i \in [n]$. In the following, we discuss two types of correctness in our proposed scheme.

1. **Normal encryption.** If gid = gid′ then, for any ciphertext computed as $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, \mathsf{att}, \boldsymbol{v})$ where att = (MFE.ct, gid′). From the correctness of ABIPFE scheme, we can write,

$$\mathsf{ABIPFE.Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle \text{ if } \mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{MFE.ct}, \mathsf{gid}') = 0$$

   which follows from the correctness of our underlying decryption algorithm of MFE scheme as $\mathsf{MFE.Dec}(\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}}, \mathsf{MFE.ct}) = 1$.

2. **Special encryption.** If gid = gid′, then for any ciphertext computed as $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, \mathsf{att}, \boldsymbol{v})$ where att = (MFE.ct, gid′). From the correctness of MFE scheme,

$$\begin{aligned} &\mathsf{MFE.Dec}(\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}}, \mathsf{MFE.SK\text{-}Enc}(\mathsf{MFE.msk}, f_{i,\ell,\beta,\mathsf{gid}'})) \\ &= \mathsf{MFE.Dec}(\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}}, \mathsf{MFE.PK\text{-}Enc}(\mathsf{MFE.mpk})) = 1 \end{aligned}$$

   i.e., $f_{i^*,\ell^*,b^*,\mathsf{gid}'}(i, \mathsf{id}, \mathsf{gid}) = 1 - \mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{MFE.ct}, \mathsf{gid}') = 1$ this implies that $\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{MFE.ct}, \mathsf{gid}') = 0$. So, from the correctness of ABIPFE, we have that with all but negligible probability,

$$\mathsf{ABIPFE.Dec}(\mathsf{sk}_{\boldsymbol{u}}, \mathsf{ct}_{\boldsymbol{v}}) = \langle \boldsymbol{u}, \boldsymbol{v} \rangle$$

   .

8.2 Security Analysis

This section shows the IND-CPA security of our EIPL-IBIPFE from LWE. To prove this, we consider the following Theorem. 8.

Theorem 10 *If the underlying ABIPFE is IND-CPA secure and the MFE scheme is secure as per Definitions 4, 5, then our EIPL-IBIPFE scheme is secure as per Definitions 13 to 17.*

Proof. We prove this theorem by using the following Lemmas 17 to Lemma 21.

Lemma 17 *If the underlying MFE scheme is 1-bounded restricted A-IND secure, then our EIPL-IBIPFE scheme achieves 1-bounded normal-hiding security as per Definition 13.*

Proof. We prove the normal-hiding security depending on the 1-bounded restricted A-IND security of MFE scheme. Let $\mathscr{B}_1$ be an A-IND adversary of the underlying MFE scheme. On receiving $(1^k, 1^{k'}, \mathsf{gid}^*, 1^m, n)$ from $\mathscr{A}$, it sets the parameter $\bar{k}$ as in the MFE scheme (which is the functionality index of the function class containing $f_{i,\ell,b,\mathsf{gid}}$) and set the challenge function $f^* = f_{(1,\perp,0,\mathsf{gid}^*)}$. By performing the following steps, $\mathscr{B}$ simulates the normal-hiding game.

Simulation of $\mathscr{B}_1(1^{\bar{k}}, f^*)$:

Public key simulation:

1. The MFE challenger creates the challenge for $\mathscr{B}$ as follows.
   1.1 Generates $(\mathsf{MFE.msk}, \mathsf{MFE.mpk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda, 1^{\bar{k}})$.
   1.2 Choose $\mathfrak{b} \leftarrow \{0,1\}$.
   1.3 Run encryption algorithm and generates $\mathsf{MFE.ct}_0 \leftarrow \mathsf{MFE.PK\text{-}Enc}(\mathsf{MFE.mpk})$, $\mathsf{MFE.ct}_1 \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{MFE.msk}, f^*)$.
   1.4 The challenger sends $(\mathsf{MFE.mpk}, \mathsf{MFE.ct}_\mathfrak{b})$ to $\mathscr{B}_1$.
2. $\mathscr{B}_1$ chooses $(\ell', \beta') \leftarrow ([k] \cup \{\perp\}) \times \{0,1\}$ and makes a SK-Enc query for $f_{(1,\ell',\beta',\mathsf{gid}^*)}$. It receives a ciphertext MFE.ct from its challenger.
3. $\mathscr{B}_1$ computes $(\mathsf{ABIPFE.msk}, \mathsf{ABIPFE.mpk}) \leftarrow \mathsf{ABIPFE.Setup}(1^\lambda, 1^m, 1^{\kappa+k'})$.
4. $\mathscr{B}_1$ sends $\mathsf{mpk} = (\mathsf{ABIPFE.mpk}, \mathsf{MFE.mpk})$ to $\mathscr{A}$.

Secret key simulation:

5. $\mathscr{A}$ can make queries $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ to KeyGen oracle.
   5.1 $\mathscr{B}_1$ sends $(i, \mathsf{id}, \mathsf{gid})$ tuple to its challenger to get MFE secret key as $\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{MFE.msk}, (i, \mathsf{id}, \mathsf{gid}))$.
   5.2 $\mathscr{B}_1$ constructs the circuit $\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot)$ and sends ABIPFE secret key as $\mathsf{ABIPFE.sk}_{\boldsymbol{u}} \leftarrow \mathsf{ABIPFE.KeyGen}(\mathsf{ABIPFE.msk}, \mathscr{C}_{i,\mathsf{id},\mathsf{gid}}, \boldsymbol{u})$ to $\mathscr{A}$.

Special encryption simulation:

6. $\mathscr{A}$ can make at most one SplEnc query for $(\boldsymbol{v}, \mathsf{gid}^*, (1, \ell, \beta))$. Then $\mathscr{B}_1$ responds to the query as follows.
   6.1 If $(\ell', \beta') \neq (\ell, \beta)$, then return $\perp$.
   6.2 $\mathscr{B}_1$ uses the attribute $(\mathsf{MFE.ct}, \mathsf{gid}^*)$ corresponding to the function $f_{(1,\ell',\beta',\mathsf{gid}^*)}$ obtain in Step 2 and computes $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}^*), \boldsymbol{v})$.
   6.3 $\mathscr{B}_1$ sends $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$ to $\mathscr{A}$.

Note 1. $\mathscr{B}_1$ needs to guess (ℓ, β) in advance successfully to answer the $\mathscr{A}$'s query. Thus, it makes a polynomial security loss of $\frac{1}{2(k+1)}$. The reduction algorithm need to guess the value ℓ, we extend the analysis to prove q-bounded EIPL-IBIPFE (adaptive) security assuming that the q-bounded restricted MFE security for a constant q. One more observation is that a q-bounded EIPL-IBIPFE selective security can be proven directly from q-bounded restricted MFE security without any security loss.

Challenge ciphertext simulation:

7. $\mathscr{A}$ now sends the challenge message $\boldsymbol{v}$ to $\mathscr{B}_1$ and $\mathscr{B}_1$ produces the respond as follows.
   7.1 $\mathscr{B}_1$ uses the attribute $(\mathsf{MFE.ct}_{\mathfrak{b}}, \mathsf{gid}^*)$ obtain in Step 1 and then computes the ciphertext $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}\,(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}_{\mathfrak{b}}, \mathsf{gid}^*), \boldsymbol{v})$.
   7.2 $\mathscr{B}_1$ sends $\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$ to $\mathscr{A}$.

Post-challenge key-query simulation:

8. $\mathscr{A}$ can repeat key generation phase, special encryption query phase if SplEnc oracle is not queried before.

Guess:

9. Finally, $\mathscr{A}$ submits a guess bit $\mathfrak{b}'$ which is then output of $\mathscr{B}_1$ for its own guess.

Analysis of simulation. Note that, $\mathscr{B}_1$ is an admissible adversary for 1-query restricted A-IND security of MFE. It correctly simulates the normal-hiding security game for the adversary $\mathscr{A}$. Thus if $\mathscr{A}'$s advantage is non-negligible then $\mathscr{B}_1$ breaks the A-IND security of MFE schemes with non-negligible advantage. □

Lemma 18 *If the underlying MFE scheme is 1-bounded restricted F-IND secure, then our EIPL-IBIPFE scheme achieves the 1-bounded index-hiding security as per Definition 14.*

Proof. We prove the index-hiding security game depending on the 1-bounded restricted F-IND security of MFE scheme. Let $\mathscr{B}_2$ be an F-IND adversary of the underlying MFE scheme. On receiving $(1^k, 1^{k'}, 1^m, 1^n, \mathsf{gid}^*, i^*)$ from $\mathscr{A}$, it sets the parameter $\bar{k}$ as in the MFE scheme and take the challenge functions as $f^{(0)} = f_{(i^*,\perp,0,\mathsf{gid}^*)}$, $f^{(1)} = f_{(i^*+1,\perp,0,\mathsf{gid}^*)}$. In the following, $\mathscr{B}_2$ performs the following steps to simulate the index-hiding game.

Simulation of $\mathscr{B}_2(1^{\bar{k}}, f^{(0)}, f^{(1)})$:

Public key simulation:

1. The MFE challenger creates the challenge for $\mathscr{B}_2$ as follows.
   1.1 Generates $(\mathsf{MFE.msk}, \mathsf{MFE.mpk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda, 1^{\bar{k}})$.
   1.2 Choose $\mathfrak{b} \leftarrow \{0,1\}$.
   1.3 Computes $\mathsf{MFE.ct}_{\mathfrak{b}} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{MFE.mpk}, f^{(\mathfrak{b})})$.
   1.4 The challenger sends $(\mathsf{MFE.mpk}, \mathsf{MFE.ct}_{\mathfrak{b}})$ to $\mathscr{B}_2$.
2. $\mathscr{B}_2$ samples uniformly $(i', \ell', \beta') \leftarrow \{i^*, i^*+1\} \times ([k] \cup \{\perp\}) \times \{0,1\}$ and makes a secret encryption SK-Enc query for $f_{i',\ell',\beta',\mathsf{gid}^*}$ and receives a ciphertext MFE.ct from its challenger.
3. $\mathscr{B}_2$ computes $(\mathsf{ABIPFE.msk}, \mathsf{ABIPFE.mpk}) \leftarrow \mathsf{ABIPFE.Setup}(1^\lambda, 1^m, 1^{\kappa+k'})$.
4. $\mathscr{B}_2$ sends $\mathsf{mpk} = (\mathsf{ABIPFE.mpk}, \mathsf{MFE.mpk})$ to $\mathscr{A}$.

Secret key simulation:

5. $\mathscr{A}$ can make queries $\{(\tau, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})\}$ with $(\tau, \mathsf{gid}) \neq (i^*, \mathsf{gid}^*)$ to KeyGen oracle.
   5.1 $\mathscr{B}_2$ sends $(\tau, \mathsf{id}, \mathsf{gid})$ tuple to its challenger to get MFE secret key as $\mathsf{MFE.sk}_{\tau,\mathsf{id},\mathsf{gid}} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{MFE.msk}, (\tau, \mathsf{id}, \mathsf{gid}))$.
   5.2 $\mathscr{B}_2$ constructs the circuit $\mathscr{C}_{\tau,\mathsf{id},\mathsf{gid}}(\cdot,\cdot)$ and sends the ABIPFE secret key to the adversary as $\mathsf{ABIPFE.sk}_{\boldsymbol{u}} \leftarrow \mathsf{ABIPFE.KeyGen}(\mathsf{ABIPFE.msk}, \mathscr{C}_{\tau,\mathsf{id},\mathsf{gid}}, \boldsymbol{u})$ to $\mathscr{A}$.

Special encryption simulation:

6. $\mathscr{A}$ can make at most one SplEnc query for $(\boldsymbol{v}, \mathsf{gid}^*, (\tau, \ell, \beta))$ with $\tau \in \{i^*, i^*+1\}$. Then $\mathscr{B}_2$ responds to the query as follows.
   6.1 If $(\tau, \ell, \beta) \neq (i', \ell', \beta')$, then return $\perp$.
   6.2 $\mathscr{B}_2$ uses the attribute $(\mathsf{MFE.ct}, \mathsf{gid}^*)$ corresponding to the function $f_{(i',\ell',\beta',\mathsf{gid}^*)}$ obtain in Step 2 and computes $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}^*), \boldsymbol{v})$.
   6.3 $\mathscr{B}_2$ sends $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$ to $\mathscr{A}$.

   Note 2. As $\mathscr{B}_2$ needs to guess the correct (τ, ℓ, β) from the set $\{i^*, i^*+1\} \times ([k] \cup \{\perp\}) \times \{0,1\}$ before the adversary $\mathscr{A}$ makes any query to SplEnc oracle. So, it makes a polynomial security loss of $\frac{1}{4(k+1)}$.

Challenge ciphertext simulation:

7. $\mathscr{A}$ now sends the challenge message vector $\boldsymbol{v}$ to $\mathscr{B}_2$ and $\mathscr{B}_2$ produces the respond as follows.
   7.1 $\mathscr{B}_2$ uses the attribute $(\mathsf{MFE.ct}_{\mathfrak{b}}, \mathsf{gid}^*)$ obtain in Step 1 and then computes ABIPFE ciphertext as $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}\ (\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}_{\mathfrak{b}}, \mathsf{gid}^*), \boldsymbol{v})$.
   7.2 $\mathscr{B}_2$ sends $\mathsf{ct}^{(\mathfrak{b})}_{\boldsymbol{v}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$ to $\mathscr{A}$.

Post-challenge secret key simulation:

8. $\mathscr{A}$ can repeat key generation phase special encryption query phase if SplEnc oracle is not queried before.

Guess:

9. Finally, $\mathscr{A}$ submits a guess $\mathfrak{b}'$ which is the output of $\mathscr{B}_2$ for its own guess.

Note that, $f^{(0)}(\tau, \mathsf{id}, \mathsf{gid}) = f^{(1)}(\tau, \mathsf{id}, \mathsf{gid})$ for all queries $(\tau, \mathsf{id}, \mathsf{gid})$ made by $\mathscr{B}_2$ in key queries. Since $(\tau, \mathsf{gid}) \neq (i^*, \mathsf{gid}^*)$ we have,

$$\left.\begin{aligned} f^{(0)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*,\perp,0,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 0 \\ f^{(1)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*+1,\perp,0,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 0 \end{aligned}\right\} \text{if } (\tau \leq i^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*)$$

$$\left.\begin{aligned} f^{(0)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*,\perp,0,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 1 \\ f^{(1)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*+1,\perp,0,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 1 \end{aligned}\right\} \text{if } (\tau > i^*) \wedge (\mathsf{gid} = \mathsf{gid}^*)$$

Analysis of simulation. Thus $\mathscr{B}_2$ is an admissible adversary for 1-query restricted F-IND security of MFE scheme. If $\mathfrak{b} = 0$ then the challenge ciphertext corresponds to the index i^* and if $\mathfrak{b} = 1$ then it corresponds to the index i^*+1. Thus, $\mathscr{B}_2$ simulates $\mathscr{A}$ except for a polynomial security loss of $\frac{1}{4(k+1)}$. Thus if $\mathscr{A}'$s advantage is non-negligible then $\mathscr{B}_2$ breaks the F-IND security of MFE schemes with non-negligible advantage. In other words, if the advantage of $\mathscr{A}$ is non-negligible ϵ in the index-hiding game, then $\mathscr{B}_2$ breaks the F-IND security of MFE with probability of $\frac{\epsilon}{4(k+1)}$. □

Lemma 19 *If the underlying MFE scheme is 1-bounded restricted F-IND secure, then our EIPL-IBIPFE scheme satisfies the 1-bounded upper identity-hiding security game as per Definition 16.*

Proof. We prove the upper identity-hiding security depending on the 1-bounded restricted F-IND security of the MFE scheme. Let $\mathscr{B}_3$ be an F-IND adversary for the underlying MFE scheme. On receiving $(1^k, 1^{k'}, 1^m, 1^n, \mathsf{gid}^*, i^*, \ell^*, b^*)$ from $\mathscr{A}$, it sets the parameter $\bar{k}$ as in the MFE scheme and sets the challenge functions as $f^{(0)} = f_{i^*+1,\perp,0,\mathsf{gid}^*}$, $f^{(1)} = f_{i^*,\ell^*,b^*,\mathsf{gid}^*}$. $\mathscr{B}_3$ performs the following simulation as follows:

Simulation of $\mathscr{B}_3(1^{\bar{k}}, f^{(0)}, f^{(1)})$:

Public key simulation:

1. The MFE challenger creates the challenge for $\mathscr{B}_3$ as follows.
   1.1 Generates $(\mathsf{MFE.msk}, \mathsf{MFE.mpk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda, 1^{\bar{k}})$.
   1.2 Choose $\mathfrak{b} \leftarrow \{0,1\}$.
   1.3 Computes $\mathsf{MFE.ct}_\mathfrak{b} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{MFE.mpk}, f^{(\mathfrak{b})})$.
   1.4 The challenger sends $(\mathsf{MFE.mpk}, \mathsf{MFE.ct}_\mathfrak{b})$ to $\mathscr{B}_3$.
2. $\mathscr{B}_3$ selects randomly $(i', \ell', \beta') \leftarrow \{i^*, i^*+1\} \times ([k] \cup \{\perp\}) \times \{0,1\}$ which makes a SK-Enc query for $f_{i',\ell',\beta',\mathsf{gid}^*}$ and receives a ciphertext MFE.ct from its challenger.
3. $\mathscr{B}_3$ computes $(\mathsf{ABIPFE.msk}, \mathsf{ABIPFE.mpk}) \leftarrow \mathsf{ABIPFE.Setup}(1^\lambda, 1^m, 1^{\kappa+k'})$.
4. $\mathscr{B}_3$ sends $\mathsf{mpk} = (\mathsf{ABIPFE.mpk}, \mathsf{MFE.mpk})$ to $\mathscr{A}$.

Secret key simulation:

5. $\mathscr{A}$ can make queries $(\tau, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ to KeyGen oracle with the restriction that, if $\tau = i^*$ then $\mathsf{id}_{\ell^*} = b^*$.
   5.1 $\mathscr{B}_3$ sends $(\tau, \mathsf{id}, \mathsf{gid})$ tuple to its challenger to get MFE secret key as $\mathsf{MFE.sk}_{\tau,\mathsf{id},\mathsf{gid}} \leftarrow \mathsf{MFE.KeyGen}(\mathsf{MFE.msk}, (\tau, \mathsf{id}, \mathsf{gid}))$.
   5.2 $\mathscr{B}_3$ constructs the circuit $\mathscr{C}_{\tau,\mathsf{id},\mathsf{gid}}(\cdot,\cdot)$ and sends the ABIPFE secret key as $\mathsf{ABIPFE.sk}_{\boldsymbol{u}} \leftarrow \mathsf{ABIPFE.KeyGen}(\mathsf{ABIPFE.msk}, \mathscr{C}_{\tau,\mathsf{id},\mathsf{gid}}, \boldsymbol{u})$ to $\mathscr{A}$.

Special encryption simulation:

6. $\mathscr{A}$ can make at most one SplEnc query for $(\boldsymbol{v}, \mathsf{gid}^*, (\tau, \ell, \beta))$ where $\tau \in \{i^*, i^*+1\}$. Then $\mathscr{B}_3$ responds to the query as follows.
   6.1 If $(\tau, \ell, \beta) \neq (i', \ell', \beta')$, then return $\perp$.
   6.2 $\mathscr{B}_3$ uses the attribute $(\mathsf{MFE.ct}, \mathsf{gid}^*)$ corresponding to the function $f_{i',\ell',\beta',\mathsf{gid}^*}$ obtained in Step 2 and computes $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}^*), \boldsymbol{v})$.
   6.3 $\mathscr{B}_3$ sends $\mathsf{ct}_{\boldsymbol{v}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$ to $\mathscr{A}$.

   Note 3. Note that, $\mathscr{B}_3$ needs to guess the correct (τ, ℓ, β) from the set $\{i^*, i^*+1\} \times ([k] \cup \{\perp\}) \times \{0,1\}$ before the adversary $\mathscr{A}$ makes any query to SplEnc. Hence, we have a polynomial security loss of $\frac{1}{4(k+1)}$.

Challenge ciphertext simulation:

7. $\mathscr{A}$ now sends the challenge message $\boldsymbol{v}$ to $\mathscr{B}_3$ and $\mathscr{B}_3$ produces the respond as follows.
   7.1 $\mathscr{B}_3$ uses the attribute $(\mathsf{MFE.ct}_\mathfrak{b}, \mathsf{gid}^*)$ obtained in Step-1 and computes ABIPFE ciphertext as $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}_\mathfrak{b}, \mathsf{gid}^*), \boldsymbol{v})$.
   7.2 $\mathscr{B}_3$ sends $\mathsf{ct}_{\boldsymbol{v}}^{(\mathfrak{b})} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}}$ to $\mathscr{A}$.

Post-challenge secret key simulation:

8. $\mathscr{A}$ can repeat key generation phase special encryption query phase if SplEnc oracle is not queried before.

Guess:

9. Finally, $\mathscr{A}$ submits a guess $\mathfrak{b}'$ which is then the output of $\mathscr{B}_3$ for its own guess.

Note that, $f^{(0)}(\tau, \mathsf{id}, \mathsf{gid}) = f^{(1)}(\tau, \mathsf{id}, \mathsf{gid})$ for all queries $(\tau, \mathsf{id}, \mathsf{gid})$ made by $\mathscr{B}_3$ in the secret key query phase should satisfy the conditions that if $(\tau, \mathsf{gid}) = (i^*, \mathsf{gid}^*)$ then $\mathsf{id}_{\ell^*} \neq b^*$.

$$\left.\begin{aligned} f^{(0)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*+1,\perp,0,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 0 \\ f^{(1)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*,\ell^*,b^*,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 0 \end{aligned}\right\} \text{ if } (\tau \leq i^*) \wedge (\mathsf{id}_{\ell^*} = \mathfrak{b}^*) \wedge (\mathsf{gid} \neq \mathsf{gid}^*);$$

$$\left.\begin{aligned} f^{(0)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*+1,\perp,0,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 1 \\ f^{(1)}(\tau, \mathsf{id}, \mathsf{gid}) &= f_{i^*,\ell^*,b^*,\mathsf{gid}^*}(\tau, \mathsf{id}, \mathsf{gid}) = 1 \end{aligned}\right\} \text{ if } (\tau > i^*)' \wedge (\mathsf{gid} = \mathsf{gid}^*);$$

Analysis of simulation. Thus, $\mathscr{B}_3$ is an admissible adversary for 1-query restricted F-IND security of MFE scheme. If $\mathfrak{b} = 0$ then the challenge message corresponds to $(i^* + 1, \perp, 0)$ and if $\mathfrak{b} = 1$ then it corresponds to the index (i^*, ℓ^*, b^*). Hence, $\mathscr{B}_3$ simulates $\mathscr{A}$ except for a polynomial security loss of $\frac{1}{4(k+1)}$. Thus if $\mathscr{A}'$s advantage is non-negligible then $\mathscr{B}_3$ breaks the F-IND security of MFE schemes with non-negligible advantage. In other words, if the advantage of $\mathscr{A}$ is non-negligible $\epsilon(\cdot)$ in the upper identity-hiding game, then $\mathscr{B}_3$ breaks the F-IND security of MFE with probability of $\frac{\epsilon}{4(k+1)}$. □

Lemma 20 *If the underlying MFE scheme is 1-bounded restricted F-IND secure, then our EIPL-IPFE scheme achieves the 1-bounded lower identity-hiding security as per Definition 15.*

Proof of above Lemma 20 is similar with the Lemma 19.

Lemma 21 *If the underlying ABIPFE scheme is IND-CPA secure, then our EIPL-IBIPFE scheme achieves 1-bounded message-hiding security as per Definition 17.*

Proof. We prove the message-hiding security based on the IND-CPA security of ABIPFE scheme. Let $\mathscr{B}_5$ be an adversary against IND-CPA of ABIPFE. On receiving $(1^k, 1^{k'}, 1^m, 1^n, i^*, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$ from $\mathscr{A}$, it sets the parameter $\bar{k}$ as the MFE scheme and simulates $\mathscr{A}$ as follows.

Simulation of $\mathscr{B}_5(1^{\bar{k}}, 1^m, 1^n, 1^k, 1^{k'}, i^*, \mathsf{gid}^*, \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$:

Public key simulation:

1. $\mathscr{B}_5$ receives ABIPFE.mpk from ABIPFE challenger.
2. It generates $(\mathsf{MFE.msk}, \mathsf{MFE.mpk}) \leftarrow \mathsf{MFE.Setup}(1^\lambda, 1^{\bar{k}})$.
3. Then $\mathscr{B}_5$ sends $\mathsf{mpk} = (\mathsf{ABIPFE.mpk}, \mathsf{MFE.mpk})$ to $\mathscr{A}$.

Secret key simulation:

4. $\mathscr{A}$ makes secret key queries for $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ to KeyGen oracle and $\mathscr{B}_5$ responds as follows.
   4.1 Generates $\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}} \leftarrow \mathsf{MEF.KeyGen}(\mathsf{MFE.msk}, (i, \mathsf{id}, \mathsf{gid}))$.
   4.2 Construct $\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot)$ using $\mathsf{MFE.sk}_{i,\mathsf{id},\mathsf{gid}}$ and sends $\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot)$ to ABIPFE challenger.
   4.3 It receives $\mathsf{ABIPFE.sk}_{\boldsymbol{u}} \leftarrow \mathsf{ABIPFE.KeyGen}(\mathsf{ABIPFE.msk}, \mathscr{C}_{i,\mathsf{id},\mathsf{gid}}, \boldsymbol{u})$ and sent it to $\mathscr{A}$.

Special encryption simulation:

5. $\mathscr{A}$ can make at most one SplEnc query for $(\boldsymbol{v}, \mathsf{gid}^*, (i^*, \ell, \gamma))$ to which $\mathscr{B}_5$ responds as follows.
   5.1 Computes $f_{i^*,\ell,\gamma,\mathsf{gid}^*}$ and generates $\mathsf{MFE.ct} \leftarrow \mathsf{MFE.SK\text{-}Enc}(\mathsf{MFE.msk}, f_{i^*,\ell,\gamma,\mathsf{gid}^*})$.
   5.2 $\mathscr{B}_5$ computes $\mathsf{ABIPFE.ct}_{\boldsymbol{v}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}), \boldsymbol{v})$ and sends it to $\mathscr{A}$.

Challenge ciphertext simulation:

6. $\mathscr{A}$ submits two challenge message vectors $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ such that for the tuple $(i, \mathsf{id}, \mathsf{gid}, \boldsymbol{u})$ queried in Step 4 we have $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$. $\mathscr{B}_5$ generates the challenge ciphertext as follows.
   6.1 $\mathscr{B}_5$ computes the function $f_{i^*,\perp,0,\mathsf{gid}^*}$ and sends the challenge attribute as $\mathsf{MFE.ct} \leftarrow \mathsf{MFE.Enc}(\mathsf{MFE.msk}, f_{i^*,\perp,0,\mathsf{gid}^*})$ (One observation is that if $\mathscr{B}_5$ sends the challenge attribute at the beginning of the experiment then we only need a selective secure ABIPFE scheme).
   6.2 $\mathscr{B}_5$ sends $((\mathsf{MFE.ct}, \mathsf{gid}^*), \boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)})$ to the ABIPFE challenger and gets ABIPFE ciphertext as $\mathsf{ABIPFE.ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{ABIPFE.Enc}(\mathsf{ABIPFE.mpk}, (\mathsf{MFE.ct}, \mathsf{gid}^*), \boldsymbol{v}^{(\mathfrak{b})})$.
   6.3 $\mathscr{B}_5$ sends the challenge ciphertext as $\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} = \mathsf{ABIPFE.ct}_{\boldsymbol{v}^{(\mathfrak{b})}}$ to the adversary.

Post-challenge secret key simulation:

7. $\mathscr{A}$ can repeat Step 4 with same restriction given in key query phase and special encryption query phase if SplEnc oracle is not queried before.

Guess:

8. Finally, $\mathscr{A}$ submits a guess $\mathfrak{b}'$ which is then the output of $\mathscr{B}_5$ for its own guess.

Analysis of simulation. First, we note that the challenge attribute $\mathsf{att} = (\mathsf{MFE.ct}, \mathsf{gid}^*)$ corresponds to $f_{i^*,\perp,0,\mathsf{gid}^*}$. Any secret key query of the form $\{\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot), \boldsymbol{u}\}$ to the ABIPFE satisfies the following:

$$\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{MFE.ct}, \mathsf{gid}^*) = \begin{cases} 1, & \text{if } (i \geq i^*) \wedge (\mathsf{gid} = \mathsf{gid}^*) \\ 0, & \text{otherwise} \end{cases}$$

Thus, for each secret-key query $(\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\cdot,\cdot), \boldsymbol{u})$ of the ABIPFE-adversary $\mathscr{B}_5$ it holds that either $\mathscr{C}_{i,\mathsf{id},\mathsf{gid}}(\mathsf{att}) = 0$ or $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$. Hence, the $\mathscr{B}_5$ is an admissible adversary of ABIPFE. Therefore, the message-hiding security of EIPL-IBIPFE follows from Sel-IND-CPA security of the ABIPFE scheme. □

This concludes the proof of the Theorem 8.2. □

References

ABCP15. Michel Abdalla, Florian Bourse, Angelo De Caro, and David Pointcheval. Simple functional encryption schemes for inner products. In *IACR International Workshop on Public Key Cryptography*, pages 733–751. Springer, 2015.

ABP+17. Shweta Agrawal, Sanjay Bhattacherjee, Duong Hieu Phan, Damien Stehlé, and Shota Yamada. Efficient public trace and revoke from standard assumptions. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 2277–2293, 2017.

ACGU20. Michel Abdalla, Dario Catalano, Romain Gay, and Bogdan Ursu. Inner-product functional encryption with fine-grained access control. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 467–497. Springer, 2020.

ADML+07. Michel Abdalla, Alexander W Dent, John Malone-Lee, Gregory Neven, Duong Hieu Phan, and Nigel P Smart. Identity-based traitor tracing. In *International Workshop on Public Key Cryptography*, pages 361–376. Springer, 2007.

AGT21. Shweta Agrawal, Rishab Goyal, and Junichi Tomida. Multi-party functional encryption. In *Theory of Cryptography Conference*, pages 224–255. Springer, 2021.

ALS16. Shweta Agrawal, Benoît Libert, and Damien Stehlé. Fully secure functional encryption for inner products, from standard assumptions. In *Annual International Cryptology Conference*, pages 333–362. Springer, 2016.

BB04. Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 2004.

BF99. Dan Boneh and Matthew Franklin. An efficient public key traitor tracing scheme. In *Annual International Cryptology Conference*, pages 338–353. Springer, 1999.

BP08. Olivier Billet and Duong Hieu Phan. Efficient traitor tracing from collusion secure codes. In *International Conference on Information Theoretic Security*, pages 171–182. Springer, 2008.

BR09. Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 407–424. Springer, 2009.

BSW06. Dan Boneh, Amit Sahai, and Brent Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 573–592. Springer, 2006.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography Conference*, pages 253–273. Springer, 2011.

BV16. Zvika Brakerski and Vinod Vaikuntanathan. Circuit-abe from lwe: unbounded attributes and semi-adaptive security. In *Annual International Cryptology Conference*, pages 363–384. Springer, 2016.

BW06. Dan Boneh and Brent Waters. A fully collusion resistant broadcast, trace, and revoke system. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 211–220, 2006.

BZ17. Dan Boneh and Mark Zhandry. Multiparty key exchange, efficient traitor tracing, and more from indistinguishability obfuscation. *Algorithmica*, 79(4):1233–1285, 2017.

CFN94. Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *Annual International Cryptology Conference*, pages 257–270. Springer, 1994.

CFNP00. Benny Chor, Amos Fiat, Moni Naor, and Benny Pinkas. Tracing traitors. *IEEE Transactions on Information Theory*, 46(3):893–910, 2000.

CPP05. Hervé Chabanne, Duong Hieu Phan, and David Pointcheval. Public traceability in traitor tracing schemes. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 542–558. Springer, 2005.

CVW+18. Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *Theory of Cryptography Conference*, pages 341–369. Springer, 2018.

DKW21. Pratish Datta, Ilan Komargodski, and Brent Waters. Decentralized multi-authority abe for dnfs from lwe. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 177–209. Springer, 2021.

DPP20. Xuan Thanh Do, Duong Hieu Phan, and David Pointcheval. Traceable inner product functional encryption. In *Cryptographers' Track at the RSA Conference*, pages 564–585. Springer, 2020.

DSP19. Edouard Dufour-Sans and David Pointcheval. Unbounded inner-product functional encryption with succinct keys. In *International Conference on Applied Cryptography and Network Security*, pages 426–441. Springer, 2019.

FNP07. Nelly Fazio, Antonio Nicolosi, and Duong Hieu Phan. Traitor tracing with optimal transmission rate. In *International Conference on Information Security*, pages 71–88. Springer, 2007.

Fre10. David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 44–61. Springer, 2010.

FT01. Amos Fiat and Tamir Tassa. Dynamic traitor tracing. *Journal of CRYPTOLOGY*, 14(3):211–223, 2001.

GGH+13. Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *Annual Cryptology Conference*, pages 479–499. Springer, 2013.

GKRW18. Rishab Goyal, Venkata Koppula, Andrew Russell, and Brent Waters. Risky traitor tracing and new differential privacy negative results. In *Annual International Cryptology Conference*, pages 467–497. Springer, 2018.

GKW18. Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 660–670, 2018.

GKW19. Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In *Theory of Cryptography Conference*, pages 149–179. Springer, 2019.

GMS12. Fuchun Guo, Yi Mu, and Willy Susilo. Identity-based traitor tracing with short private key and short ciphertext. In *European Symposium on Research in Computer Security*, pages 609–626. Springer, 2012.

JLS21. Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 60–73, 2021.

KD98. Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 145–157. Springer, 1998.

KY02a. Aggelos Kiayias and Moti Yung. Traitor tracing with constant transmission rate. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 450–465. Springer, 2002.

KY02b. Kaoru Kurosawa and Takuya Yoshida. Linear code implies public-key traitor tracing. In *International Workshop on Public Key Cryptography*, pages 172–187. Springer, 2002.

LLW21. Qiqi Lai, Feng-Hao Liu, and Zhedong Wang. New lattice two-stage sampling technique and its applications to functional encryption–stronger security and smaller ciphertexts. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 498–527. Springer, 2021.

LPSS17. San Ling, Duong Hieu Phan, Damien Stehlé, and Ron Steinfeld. Hardness of k-lwe and applications in traitor tracing. *Algorithmica*, 79(4):1318–1352, 2017.

LV16. Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 11–20. IEEE, 2016.

NWZ16. Ryo Nishimaki, Daniel Wichs, and Mark Zhandry. Anonymous traitor tracing: how to embed arbitrary information in a key. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 388–419. Springer, 2016.

PD21. Tapas Pal and Ratna Dutta. Attribute-based access control for inner product functional encryption from lwe. In *International Conference on Cryptology and Information Security in Latin America*, pages 127–148. Springer, 2021.

PSNT06. Duong Hieu Phan, Reihaneh Safavi-Naini, and Dongvu Tonien. Generic construction of hybrid public key traitor tracing with full-public-traceability. In *International Colloquium on Automata, Languages, and Programming*, pages 264–275. Springer, 2006.

PT11. Duong Hieu Phan and Viet Cuong Trinh. Identity-based trace and revoke schemes. In *International Conference on Provable Security*, pages 204–221. Springer, 2011.

SSW01. Jessica N Staddon, Douglas R Stinson, and Ruizhong Wei. Combinatorial properties of frameproof and traceability codes. *IEEE transactions on information theory*, 47(3):1042–1049, 2001.

SW98. Douglas R Stinson and Ruizhong Wei. Combinatorial properties and constructions of traceability schemes and frameproof codes. *SIAM Journal on Discrete Mathematics*, 11(1):41–53, 1998.

TT01. Wen-Guey Tzeng and Zhi-Jia Tzeng. A public-key traitor tracing scheme with revocation using dynamic shares. In *International Workshop on Public Key Cryptography*, pages 207–224. Springer, 2001.

Wat05. Brent Waters. Efficient identity-based encryption without random oracles. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 114–127. Springer, 2005.

Zha21. Mark Zhandry. White box traitor tracing. In *Annual International Cryptology Conference*, pages 303–333. Springer, 2021.

A Security Analysis of Tracing

In this section, we present the security analysis of tracing of Theorem 4.

Proof. **Correctness of Tracing.** Next, we show that the false trace probability is bounded by a negligible function, and the correct trace probability is close to the probability of $\mathscr{A}$ outputting an ϵ-successful decoding box for some non-negligible $\epsilon(\cdot)$. This proof technique is inspired from the Goyal et al. [GKW19] tracing mechanism.

Let us consider the following notations for the further proof of this Theorem. Given any pirate decoder box $\mathscr{D}$ and messages $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ for all $i \in [n+1], \ell \in [k]$, suppose

$$p^{\mathscr{D}}_{i,\perp} = \Pr\left[\mathscr{D}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{c}\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{EIPL\text{-}IPFE.SplEnc}(\mathsf{key}, \mathsf{gid}, \boldsymbol{v}^{(\mathfrak{b})}, (i, \perp, 0)),\\ \mathfrak{b} \leftarrow \{0,1\}\end{array}\right]$$

$$p^{\mathscr{D}}_{i,\ell} = \Pr\left[\mathscr{D}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{c}\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{EIPL\text{-}IPFE.SplEnc}(\mathsf{key}, \mathsf{gid}, \boldsymbol{v}^{(\mathfrak{b})}, (i, \ell, 0)),\\ \mathfrak{b} \leftarrow \{0,1\}\end{array}\right]$$

$$p^{\mathscr{D}}_{\mathsf{nrml}} = \Pr\left[\mathscr{D}(\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}}) = \mathfrak{b} : \begin{array}{c}\mathsf{ct}_{\boldsymbol{v}^{(\mathfrak{b})}} \leftarrow \mathsf{EIPL\text{-}IPFE.Enc}(\mathsf{mpk}, \mathsf{gid}, \boldsymbol{v}^{(\mathfrak{b})}),\\ \mathfrak{b} \leftarrow \{0,1\}\end{array}\right]$$

The above probabilities are computed depending on the randomness used in the special encryption.

False Trace Probability. Now we will prove that probability of the false tracing by the Trace algorithm is negligible. Now, we prove the following Lemma.

Lemma 22 *If the scheme EIPL-IBIPFE is a 1-query secure as per Definitions 13 to 17, then for every PPT adversary $\mathscr{A}$, polynomial $q(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > 1/q(\lambda)$,*

$$\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathscr{A},\epsilon}(\lambda) \leq \mathsf{negl}(\lambda)$$

, where $\mathsf{Pr\text{-}Fal\text{-}Tr}_{\mathscr{A},\epsilon}(\cdot)$ *is defined in Definition 12.*

Proof. Let $S \subseteq [n] \times \{0,1\}^k \times \{0,1\}^{k'} \times \mathscr{X}$ be the set of query tuples by the adversary $\mathscr{A}$ for secret keys and S_{index} be the set of indices queried by the adversary $\mathscr{A}$ for secret keys, and let $\mathscr{D}$ be the decoder box output by $\mathscr{A}$.
For $i \in [n], \ell \in [k]$ and $\mathsf{gid} = \mathsf{gid}^*$, we define events

$$A_i^{\mathscr{D}} : p_{i,\perp}^{\mathscr{D}} - p_{i+1,\perp}^{\mathscr{D}} > \epsilon/8n$$
$$B_{i,\ell,\mathsf{lwr}}^{\mathscr{D}} : p_{i,\perp}^{\mathscr{D}} - p_{i,\ell}^{\mathscr{D}} > \epsilon/16n$$
$$C_{i,\ell,\mathsf{upr}}^{\mathscr{D}} : p_{i,\ell}^{\mathscr{D}} - p_{i+1,\perp}^{\mathscr{D}} > \epsilon/16n$$

$$\mathsf{Diff\text{-}Adv}^{\mathscr{D}} : \bigvee_{i \in [n] \setminus S_{\mathsf{index}}} A_i^{\mathscr{D}} \bigvee_{(i,\mathsf{id}) \in S, \ell \in [k] \text{ s.t. } \mathsf{id}_\ell = 1} B_{i,\ell,\mathsf{lwr}}^{\mathscr{D}} \bigvee_{(i,\mathsf{id}) \in S, \ell \in [k] \text{ s.t. } \mathsf{id}_\ell = 0} C_{i,\ell,\mathsf{upr}}^{\mathscr{D}}$$

For simplicity of notations, we will drop dependence on decoder $\mathscr{D}$ whenever clear from context. Next, note that the probability of the event *false trace* can be rewritten (using union bound) as follows by conditioning on the events defined above

$$\mathsf{Pr}[\mathsf{Fal\text{-}Tr}] \leq \mathsf{Pr}[\mathsf{Fal\text{-}Tr} | \overline{\mathsf{Diff\text{-}Adv}}] + \sum_{i \in [n]} \mathsf{Pr}[i \notin S_{\mathsf{index}} \wedge A_i]$$
$$+ \sum_{(i,\ell) \in [n] \times [k]} \mathsf{Pr}\left[\exists \mathsf{id} \in \{0,1\}^k \text{ s.t. } (i,\mathsf{id}) \in S \wedge \begin{pmatrix} (B_{i,\ell,\mathsf{lwr}} \wedge \mathsf{id}_\ell = 1) \\ \vee (C_{i,\ell,\mathsf{upr}} \wedge \mathsf{id}_\ell = 0) \end{pmatrix} \right]$$

Now, we will show that each term is bounded by a negligible function.

Claim 10. For every PPT adversary $\mathscr{A}$, there exists a negligible function $\mathsf{negl}_1(\cdot)$ such that for all $\lambda \in \mathbb{N}$, $\mathsf{Pr}[\mathsf{Fal\text{-}Tr} | \overline{\mathsf{Diff\text{-}Adv}}] \leq \mathsf{negl}_1(\lambda)$.

Proof. Here we give a high level sketch of proof. The proof follows from Chernoff bounds and similar to Lemma 4.4 of [GKW18], and Lemma 5.3 of [GKRW18]. Note that the tracing algorithm outputs a user identity which was not allowed to key query by the adversary iff the event Fal-Tr occurs. Recall that the tracing algorithm first trace the key indices of the corrupted keys then trace the corresponding identities. There are two sources of error in incorrect tracing. First, during step one of tracing the algorithm might incorrectly include some index $i \notin S_{\mathsf{index}}$ in the traitor's index-set T^{index}. In the second phase of the tracing procedure, it may happen that this outputs a non-corrupt identity id for some index $i \in S_{\mathsf{index}}$, that is for some $i \in S_{\mathsf{index}}$ the ID-Trace algorithm traces the id incorrectly at least one bit position. By using the union bound, we can represent it as follows: (recall that T and T^{index} are introduced in the description of Trace algorithm)

$$\mathsf{Pr}[\mathsf{Fal\text{-}Tr} | \overline{\mathsf{Diff\text{-}Adv}}] \leq \sum_{i \in [n]} \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \wedge i \notin S_{\mathsf{index}} \wedge (\exists p, q : (i,p,q) \in T^{\mathsf{index}}) | \overline{\mathsf{Diff\text{-}Adv}}]$$
$$+ \sum_{(i,l) \in [n] \times [k]} \mathsf{Pr}[\mathsf{Fal\text{-}Tr} \wedge \exists \mathsf{id}, \widetilde{\mathsf{id}} : (\mathsf{id}, \widetilde{\mathsf{id}}) \in S \wedge \widetilde{\mathsf{id}} \in T \wedge \mathsf{id}_\ell \neq \widetilde{\mathsf{id}}_\ell | \overline{\mathsf{Diff\text{-}Adv}}].$$

In the above inequality, the first term on the right side bounds the type 1 error (i.e., faulty step one tracing) and the second term bounds the type 2 error (i.e., faulty step two tracing). Now we explictly discuss about the first term. Note that, if event

$$\overline{\mathsf{Diff\text{-}Adv}} \text{ occurs } \implies \forall i \notin S_{\mathsf{index}}, \overline{A}_i \text{ occurred}$$

This, it must hold that for every $i \in [n]$

$$\Pr[i \notin S_{\mathsf{index}} \wedge (\exists p, q : (i, p, q) \in T^{\mathsf{index}}) | \overline{\mathsf{Diff\text{-}Adv}}] \leq 2^{-O(\lambda)}.$$

Using Chernoff bound we can argue that $\overline{A}_i$ provides that $p_{i,\perp} - p_{i+1,\perp} \leq \epsilon/8n$ and event $(\exists p, q : (i, p, q) \in T^{\mathsf{index}})$ suggest that $\hat{p}_{i,\perp} - \hat{p}_{i+1,\perp} > \epsilon/4n$ where $\hat{p}$ denotes the corresponding estimate computed by the tracing algorithm.

We now concentrate the second term. For a fixed index-position pair (i, ℓ) corresponding a particular event where the ID-Trace algorithm outputs identity of a traitor $\widetilde{\mathsf{id}}_\ell (\neq \mathsf{id}_\ell)$. Here, the adversary can able to query for the secret keys associated to the index-position pair (i, id). Note that, in each index position $\mathscr{A}$ is allowed to ask secret only one at a time. Therefore, by conditioning on the event $\overline{\mathsf{Diff\text{-}Adv}}$ we get that for every $(i, \mathsf{id}) \in S, \ell \in [k]$, event $\overline{\mathsf{Diff\text{-}Adv}_{i,\ell,X}}$ always occurs where $X = \mathsf{lwr}$ if $\mathsf{id}_\ell = 1$ else $X = \mathsf{upr}$.

Therefore, for all $(i, \mathsf{id}) \in S, \ell \in [k]$ the following probability always satisfy:

$$\Pr[\exists \mathsf{id}, \widetilde{\mathsf{id}} : (\mathsf{id}, \widetilde{\mathsf{id}}) \in S \wedge \widetilde{\mathsf{id}} \in T \wedge \mathsf{id}_\ell \neq \widetilde{\mathsf{id}}_\ell | \overline{\mathsf{Diff\text{-}Adv}}] \leq 2^{-O(\lambda)}.$$

If we assume that (i, id, ℓ) and let $\mathsf{id}_\ell = 1$. From the Chernoff bound the above inequality holds as since we know that event $\overline{B}_i$ occurs thus we have $\hat{p}_{i,\perp} - \hat{p}_{i,\ell} \leq \epsilon/16n$ and the event $\widetilde{\mathsf{id}} \in T \wedge \mathsf{id}_\ell = 0$ suggests that $\hat{p}_{i,\perp} - \hat{p}_{i,\ell} > \epsilon/8n$. Therefore, combining all the above inclusion we get that

$$\Pr[\mathsf{Fal\text{-}Tr} | \overline{\mathsf{Diff\text{-}Adv}}] \leq n \cdot 2^{-O(\lambda)} + n \cdot k \cdot 2^{-O(\lambda)} = \mathsf{negl}_1(\lambda)$$

□

Claim 11. If our EIPL-IBIPFE is a 1-query index hiding secure as per Definition 14, then for every PPT adversary $\mathscr{A}$, polynomial $q(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists a negligible function $\mathsf{negl}_2(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > 1/q(\lambda)$ and $i \in [n]$,

$$\Pr[i \notin S_{\mathsf{index}} \wedge \mathsf{Diff\text{-}Adv}_i] \leq \mathsf{negl}_2(\lambda),$$

where n is the index bound chosen, and S_{index} is the set of indices queried by $\mathscr{A}$.

Proof. The proof of this claim follows from the Lemma 4.5 of [GKW18] and Lemma 5.4 of [GKRW18]. □

Claim 12. If our EIPL-IBIPFE scheme is a 1-query lower and upper identity hiding secure as per Definitions 15 and 16, then for every PPT adversary $\mathscr{A}$, polynomial $q(\cdot)$ and non-negligible function $\epsilon(\cdot)$, there exists a negligible function $\mathsf{negl}_3(\cdot)$ such that for all $\lambda \in \mathscr{N}$ satisfying $\epsilon(\lambda) > 1/q(\lambda)$ and $i \in [n]$, $\ell \in [k]$,

$$\Pr\left[\exists \mathsf{id} \in \{0,1\}^k \text{ s.t. } (i, \mathsf{id}) \in S \wedge \begin{pmatrix} (B_i \wedge \mathsf{id}_\ell = 1) \vee \\ (C_i \wedge \mathsf{id}_\ell = 0) \end{pmatrix}\right] \leq \mathsf{negl}_3(\lambda).$$

where n is the index bound chosen, and S is the set of all queried tuple by $\mathscr{A}$ in of the form $(i.\mathsf{id}, \mathsf{gid}, \boldsymbol{u})$.

Proof. Suppose there exists a PPT adversary $\mathscr{A}$, polynomial $q(\cdot)$ and non-negligible function $\epsilon(\cdot)$, $\delta(\cdot)$ such that for all $\lambda \in \mathbb{N}$ satisfying $\epsilon(\lambda) > 1/q(\lambda)$ and $i' \in [n]$, $\ell' \in [k]$,

$$\Pr\left[\exists \mathsf{id} \in \{0,1\}^k \text{ s.t. } (i,\mathsf{id}) \in S \wedge \begin{pmatrix} (B_{i',\ell',\mathsf{lwr}} \wedge \mathsf{id}_{\ell'} = 1) \vee \\ (C_{i',\ell',\mathsf{upr}} \wedge \mathsf{id}_{\ell'} = 0) \end{pmatrix}\right] \geq \delta(\lambda).$$

Then we can use $\mathscr{A}$ to build a PPT reduction algorithm $\mathscr{B}$ that breaks the upper/lower identity hiding security property of EIPL-IPFE. The reduction algorithm $\mathscr{B}$ first receives the challenge group identity gid^* with the challenge message vector pair $\boldsymbol{v}^{(0)}, \boldsymbol{v}^{(1)}$ from the adversary and forwards it to EIPL-IBIPFE challenger. Then it gets $1^n, 1^k$ from the adversary. It chooses a index $i \leftarrow [n]$, position $\ell \in [k]$, and bit $b \in \{0,1\}$, and sends the challenge index-position-bit tuple $(i,\ell,0)$ and $(1^n, 1^k, 1^{k'}, 1^m)$ to the EIPL-IBIPFE challenger. It then receives the EIPL-IBIPFE public key pk from the challenger, which it sends to $\mathscr{A}$ (the reduction algorithm randomly guess (i',ℓ') as well as $b' = 0$ it interacts with EIPL-IBIPFE challenger, otherwise if $b = 1$ it interacts with EIPL-IBIPFE upper identity-hiding game). Then $\mathscr{A}$ makes secret key queries for the tuple $(j,\mathsf{id},\mathsf{gid},\boldsymbol{u})$, if $j = i^*$, $\mathsf{id}_\ell = b^*$ and $\mathsf{gid} = \mathsf{gid}^*$, then $\mathscr{B}$ aborts and sends a random bit as guess bit to the EIPL-IBIPFE challenger. Else, on key query for $(j,\mathsf{id},\mathsf{gid},\boldsymbol{u})$ from $\mathscr{A}$, the reduction algorithm $\mathscr{B}$ forwards $(j,\mathsf{id},\mathsf{gid},\boldsymbol{u})$ to the EIPL-IBIPFE challenger, if $\mathsf{gid} = \mathsf{gid}^*$, EIPL-IBIPFE's challenger generates his response and sends it to $\mathscr{B}$ then it forwards the challengers response to the adversary. After all key queries, the adversary outputs a decoding box $\mathscr{D}_{\boldsymbol{u}}$ to $\mathscr{B}$ and then $\mathscr{B}$ chooses two random bits α, β. Then, $\mathscr{B}$ sends message $\boldsymbol{v}^{(\alpha)}$ as its challenge message, and receives challenge ciphertext ct^* from EIPL-IBIPFE challenger. It also queries the EIPL-IBIPFE challenger for a special-encryption of $\boldsymbol{v}^{(\alpha)}$ to the index-position-value tuple $(i,\ell,0)$ if $\beta = 0$, else for $(i+b,\perp,0)$. Let ct be the challenger's response. Finally, $\mathscr{B}$ runs decoder box $\mathscr{D}_{\boldsymbol{u}}$ on ct and ct^* independently, and if $\mathscr{D}_{\boldsymbol{u}}(\mathsf{ct}) = \mathsf{ct}^*$, it outputs $b' = \beta$, else it outputs $b' = 1-\beta$ as it guess. Since in the upper identity/lower identity-hiding security $\mathscr{B}$ is an admissible adversary, in other words if $b = 0$ then lower identity-hiding else it achieves upper identity-hiding security respectively. As $\mathscr{B}$ does not query for the secret key corresponding to the tuple $(j,\mathsf{id},\mathsf{gid},\boldsymbol{u})$ such that $\mathsf{id}_\ell = b$. Also, $\mathscr{B}$ can make only one query to the special encryption oracle to the index-position-bit tuple $(i,\ell,0)$ and $(i+b,\perp,0)$. Therefore, from the Lemma 4.1 and 4.5 of [GKW18] and Lemma 5.4 of [GKRW18], we compute the advantage of the reduction algorithm is at least $\frac{1}{2kn} \cdot (\frac{\epsilon}{16n})^2$. Thus the claim follows. □

Therefore, it follows that the probability of the false trace is at most $\mathsf{negl}(\lambda) + n \cdot \mathsf{negl}_2(\lambda) + nk \cdot \mathsf{negl}(\lambda)$. □

Correct trace probability. In the following, we show that if the adversary outputs a good decoder, then the tracing algorithm outputs a non-empty set T with negligible probability. The correctness of the tracing follows from the Lemma 22. We provide a formal reduction for the correct trace in the following.

Lemma 23 *If our EIPL-IBIPFE is 1-query secure as per Definitions 13 to 17. Then for every PPT adversary $\mathscr{A}$, polynomial $q(\cdot)$, there exist a negligible function negl such that $\lambda \in \mathbb{N}, \epsilon(\lambda) > 1/q(\lambda)$ such that*

$$\Pr\text{-}\mathsf{Cor}\text{-}\mathsf{Tr}_{\mathscr{A},\epsilon}(\lambda) \geq \Pr\text{-}\mathsf{G}\text{-}\mathsf{D}_{\mathscr{A},\epsilon}(\lambda) - \mathsf{negl}(\lambda)$$

where $\Pr\text{-}\mathsf{Cor}\text{-}\mathsf{Tr}_{\mathscr{A},\epsilon}(\cdot)$ *and* $\Pr\text{-}\mathsf{G}\text{-}\mathsf{D}_{\mathscr{A},\epsilon}(\lambda)$ *are defined in Definition 12.*

Proof. First we analysis that teh tracing algorithm outputs a non-empty index set T. As it is known that if the event Good-Decoder occurs occurs then $p^{\mathscr{D}}_{\mathsf{nrml}}(\lambda) \geq 1/2 + \epsilon$ for some non-negligible function $\epsilon(\cdot)$. Let $S_{\mathsf{index}} \subseteq$ with $p^{\mathscr{D}}_{i,\perp} - p^{\mathscr{D}}_{i+1,\perp}.1/2 + \lambda$. From Claim 10, for all $i \in S_{\mathsf{index}}$

we have

$$\Pr[\hat{p}^{\mathscr{D}}_{i,\perp} - \hat{p}^{\mathscr{D}}_{i+1,\perp} < \frac{\epsilon}{4n}] \leq \frac{1}{2^{O(\lambda)}} \text{ [using Chernoff bound]} \tag{31}$$

where $\hat{p}$ represents the corresponding estimate evaluated by tracing algorithm. From the message-hiding and normal-hiding security, we have

$$\text{for } \tau \geq i, \;\; p^{\mathscr{D}}_{\tau,\perp} \leq 1/2 + \mathsf{negl}_2(\lambda), \quad p^{\mathscr{D}}_{\mathsf{nrml}} - p^{\mathscr{D}}_{1,\perp} \leq \mathsf{negl}_3(\lambda).$$

Recall that in the message hiding security game, all key queries for the tuple $(\tau.\mathsf{id}, \mathsf{gid}^*, \boldsymbol{u})$ must satisfy the fact that $\langle \boldsymbol{u}, \boldsymbol{v}^{(0)} \rangle = \langle \boldsymbol{u}, \boldsymbol{v}^{(1)} \rangle$ for all $\tau \geq i$. For some negligible functions negl_2, negl_3. Therefore, we can write

$$p^{\mathscr{D}}_{1,\perp} - p^{\mathscr{D}}_{\tau,\perp} \geq \epsilon - \mathsf{negl}_2(\lambda) - \mathsf{negl}_3(\lambda) > \epsilon/2$$

Therefore, $S_{\mathsf{index}} \neq \phi$ whenever the event Good-Decoder occurs. So from the above Equation 31, it can be concluded that whenever Good-Decoder occurs then

$$T^{\mathsf{index}} \neq \phi \wedge [\forall (i,p,q) \in T^{\mathsf{index}} : p - q > \epsilon/4n]$$

Therefore, for every (i,p,q) tuple, from ID-Trace algorithm it outputs some identity id. Since for all $\ell \in [k]$, either $p^{\mathscr{D}}_{i,\ell} > (p+q)/2$ then the algorithm put $\mathsf{id}_\ell = 1$ otherwise, it outputs $\mathsf{id}_\ell = 0$. Therefore, $T^{\mathsf{index}} \neq \phi \implies T \neq \phi$. Hence, it follows that

$$\Pr[T \neq \phi] \geq (1 - n \cdot \mathsf{negl}_1(\lambda)) \cdot \Pr\text{-G-D}_{\mathscr{A},\epsilon}(\lambda) \geq \Pr\text{-G-D}_{\mathscr{A},\epsilon}(\lambda) - \mathsf{negl}(\lambda).$$

From Lemma 22, we conclude that

$$\Pr\text{-Cor-Tr}_{\mathscr{A},\epsilon}(\lambda) \geq \Pr\text{-G-D}_{\mathscr{A},\epsilon}(\lambda) - \mathsf{negl}(\lambda)$$

This conclude the proof. □

The concludes the proof of EI-TIBIPFE security Theorem 2 follows. □

B IBIPFE from DBDH

In this section, we present our construction of IBIPFE from the DBDH assumption. Technically, we extend the framework of Water's IBE [Wat05] and add inner product functionality into it for building our IBIPFE. It consists of four PPT algorithms IBIPFE = (Setup, KeyGen, Enc, Dec) and details about these algorithms are given below.

Setup$(1^\lambda, 1^m, 1^{k'}) \to$ (msk, mpk): On input the security parameter λ, a vector length m and identity space parameter k', the trusted authority generates a master secret key msk and a master public key mpk. The algorithm performs as follows:

- Consider a bilinear group $\mathbb{BG} \leftarrow \mathscr{G}_{\mathsf{BG.Gen}}(1^\lambda)$ where $\mathbb{BG} = (p, g, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ and $\mathbb{G}$ is a prime order group with order p.
- Chooses a random generator $g \in \mathbb{G}$ and a random group element g_2 from the group $\mathbb{G}$.
- Sets $g_{1,i} = g^{\alpha_i}$ and $g_{2,i} = g_2^{\alpha_i}$ for all $i \in [m]$, where α_i's are the random exponents chosen from the field $\mathbb{Z}_p$.
- Additionally, the authority chooses a random group element $u' \in \mathbb{G}$ and a random k'-length vector $\boldsymbol{u} = (u_i) \in \mathbb{G}^{k'}$ whose elements are chosen at random from $\mathbb{G}$.
- Sets the master public key $\mathsf{mpk} = \left(\{g_{1,i}\}_{i=1}^m, g_2, u', \boldsymbol{u}, g\right)$ and the master secret key $\mathsf{msk} = \left(\{g_{2,i}\}_{i=1}^m\right)$.

KeyGen(msk, gid, $\boldsymbol{y}$) $\rightarrow \mathsf{sk}_{\boldsymbol{y}}$: The trusted authority takes the master secret key msk, a vector $\boldsymbol{y} \in \mathbb{Z}^m$ and an identity gid $\in \mathscr{ID}$ as input and outputs a secret key $\mathsf{sk}_{\boldsymbol{y}}$ corresponding to the vector $\boldsymbol{y}$ and identity gid. This algorithm performs as follows:

- Let gid be an k'-bit string representing an identity, where gid_i denotes the i-th bit of gid and $\mathcal{V} \subseteq \{1, 2, \ldots, k'\}$ be set of all i for which $\mathsf{gid}_i = 1$. We consider an identity encoding function $\mathrm{H} : \mathscr{ID} \rightarrow \mathbb{G}$ be defined as $\mathrm{H}(\mathsf{gid}) = u' \prod_{j \in \mathcal{V}} u_i$ for gid $\in \mathscr{ID}$ where $\mathscr{ID}$ be the set of identities.
- Chooses $r \leftarrow \mathbb{Z}_p$.
- Outputs the secret key $\mathsf{sk}_{\boldsymbol{y}} = (d_1, d_2)$ where $d_1 = g_2^{\langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle} \mathrm{H}(\mathsf{gid})^r$, $d_2 = g^r$

Enc(mpk, gid$'$, $\boldsymbol{x}$) $\rightarrow \mathsf{ct}_{\boldsymbol{x}}$: On input as master public key mpk, a message vector $\boldsymbol{x} \in \mathbb{Z}^m$. an identity gid$'$, the encryption algorithm executes the following steps:

- Choose a random value t from $\mathbb{Z}_p$.
- Outputs the ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ as

$$\mathsf{ct}_{\boldsymbol{x}} = \left(\boldsymbol{C}_1 = e(g, g_2)^{\boldsymbol{\alpha} t + \boldsymbol{x}}, C_2 = g^t, C_3 = \mathrm{H}(\mathsf{gid}')^t \right)$$

Dec(sk$_{\boldsymbol{y}}$, ct$_{\boldsymbol{x}}$) $\rightarrow \perp$ or $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$: The decryptor uses his secret key $\mathsf{sk}_{\boldsymbol{y}}$ to decrypt the ciphertext $\mathsf{ct}_{\boldsymbol{x}}$ and outputs a decrypted value ζ or a symbol $\perp$ indicating failure. Decryptor computes

$$\zeta = \langle \boldsymbol{C}_1, \boldsymbol{y} \rangle \cdot \frac{e(d_2, C_3)}{e(d_1, C_2)}$$

Correctness. If gid = gid$'$, then correctness holds as follows:

$$\begin{aligned}
\zeta &= \langle \boldsymbol{C}_1, \boldsymbol{y} \rangle \cdot \frac{e(d_2, C_3)}{e(d_1, C_2)} \\
&= e(g, g_2)^{\langle \boldsymbol{\alpha} t + \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot \frac{e(g^r, \mathrm{H}(\mathsf{gid})^t)}{e(g_2^{\langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle} \mathrm{H}(\mathsf{gid})^r, g^t)} \\
&= e(g, g_2)^{t \langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle} \cdot e(g, g_2)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle} \cdot \frac{e(g, \mathrm{H}(\mathsf{gid}))^{rt}}{e(g_2, g)^{t \langle \boldsymbol{\alpha}, \boldsymbol{y} \rangle} e(\mathrm{H}(\mathsf{gid}), g)^{rt}} \\
&= e(g, g_2)^{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}
\end{aligned}$$

B.1 Security Analysis

In this subsection, we discuss the selective security of our IBIPFE scheme in the standard model.

Theorem 11 *If the plain DBDH assumption 2 holds over the bilinear group $\mathbb{BG}$, then our IBIPFE scheme is selective secure as per the Definition 2.*

Proof. Suppose $\mathscr{A}$ be a PPT adversary against the selective security of our IBIPFE scheme. We construct an algorithm $\mathscr{B}$ for breaking the DBDH assumption that uses $\mathscr{A}$ as a subroutine. To prove Theorem 11, we consider two games. The first game is the same as the original selective security game of IBIPFE as per Definition 2. In the next game, we change the distribution of the master public key, secret keys, and the challenge ciphertext where we first sample a random vector $\widetilde{\boldsymbol{\alpha}}$ and set $\boldsymbol{\alpha} = \mathbf{F}^\top \widetilde{\boldsymbol{\alpha}}$. The matrix $\mathbf{F}$ is a full rank matrix chosen such that $\mathbf{F}(\boldsymbol{x}^{(0)} - \boldsymbol{x}^{(1)}) = (1, 0, \cdots, 0)^\top$ where $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}$ are the challenge message vectors submitted by the $\mathscr{A}$. Assuming DBDH holds in the bilinear group $\mathbb{BG}$, we show that the adversary has a negligible advantage in distinguishing between the challenge ciphertexts.

Game 0: This game is exactly same as selective security of IBIPFE.

Game 1: The game is same as the previous game except for each identity, the challenger samples the master secret key msk as follows:

(a) Samples uniformly random vector $\widetilde{\boldsymbol{\alpha}} = (\widetilde{\alpha}_1, \widetilde{\alpha}_2, \cdots, \widetilde{\alpha}_m)$ for each $\widetilde{\alpha}_i \in \mathbb{Z}$.
(b) Samples a full rank matrix $\mathbf{F} \in \mathbb{Z}_q^{m\times m}$ satisfying the relation $\mathbf{F}(\boldsymbol{x}^{(0)} - \boldsymbol{x}^{(1)}) = (1, 0, \cdots, 0)^\top$ where $\boldsymbol{x}^{(0)}, \boldsymbol{x}^{(1)}$ are the challenge message vectors of length m.
(c) Sets $\boldsymbol{\alpha} = \mathbf{F}^\top \widetilde{\boldsymbol{\alpha}}$ instead of sampling uniformly random as in the Game 0.

In adversary's view, the master public key mpk, secret key $\mathsf{sk}_{\boldsymbol{y}}$ is associated to a vector $\boldsymbol{y}$ with an identity gid and the challenge ciphertext $\mathsf{ct}_{\boldsymbol{x}^{(\mathfrak{b})}} \leftarrow \mathsf{Enc}(\mathsf{mpk}, \mathsf{gid}^*, \boldsymbol{x}^{(\mathfrak{b})})$ are simulated as follows:

Public key: $\mathsf{mpk} = g^{\mathbf{F}^\top \widetilde{\boldsymbol{\alpha}}}$.
Secret key: For secret key query corresponding to the identity gid and predicate vector $\boldsymbol{y}$, we consider $\mathcal{V} \subseteq \{1, \cdots, k'\}$ be the set of all i for which $\mathsf{gid}_i = 1$. Then, the secret key $\mathsf{sk}_{\boldsymbol{y}} = \left(g_2^{\langle \widetilde{\boldsymbol{\alpha}}, \mathbf{F}\boldsymbol{y}\rangle} \mathrm{H}(\mathsf{gid})^r, g^r\right)$
Challenge ciphertext: For challenge identity gid^*, let $\mathcal{V}^* \subseteq \{1, \cdots, k'\}$ be the set of all i for which $\mathsf{gid}_i^* = 1$.

$$C_1 = e(g, g_2)^{\boldsymbol{\alpha} t + \mathfrak{b}(\boldsymbol{x}^{(1)} - \boldsymbol{x}^{(0)}) + \boldsymbol{x}^{(0)}} \qquad C_2 = g^t$$
$$= e(g, g_2)^{\mathbf{F}^\top(\widetilde{\boldsymbol{\alpha}} t - \mathfrak{b}\boldsymbol{e}_1 + \mathbf{F}\boldsymbol{x}^{(0)})} \qquad C_3 = \mathrm{H}(\mathsf{gid}^*)^t$$

Since, $\mathbf{F} \in \mathbb{Z}_q^{m\times m}$ is an orthogonal matrix, then the following two distributions are identical.

$$\{\boldsymbol{\alpha} : \boldsymbol{\alpha} \leftarrow \mathbb{Z}_q^m\} \equiv \{\mathbf{F}^\top \widetilde{\boldsymbol{\alpha}} : \widetilde{\boldsymbol{\alpha}} \leftarrow \mathbb{Z}_q^m\}$$

Therefore, the advantage of any PPT adversary $\mathscr{A}$ in distinguishing between Game 0 and Game 1 is negligible in the security parameter λ.

Without loss of generality, we assume that the adversary makes maximum Q number of secret keys queries and the challenge identity gid^*. In this case, the simulator chooses a random integer $\widehat{k} \leftarrow [k']$ and sets an integer $s = 10Q$. Then, it chooses a random k'-length vector $\boldsymbol{z} = (z_i) \leftarrow \mathbb{Z}_s^{k'}$ and a value $z' \leftarrow \mathbb{Z}_s$. Additionally, simulator also chooses a random value $w' \leftarrow \mathbb{Z}_p$ and a random k'-length vector $\boldsymbol{w} = (w_i) \leftarrow \mathbb{Z}_p^{k'}$. All these values are kept secret to the simulator.

Let us consider $\mathcal{V}^* \subseteq \{1, 2, \ldots, k'\}$ be the set of all i for which the challenge identity $\mathsf{gid}_i^* = 1$. Let $\mathcal{V}^* = \{i_1, i_2, \ldots, i_\kappa\}$. Now, we choose the z_i values from $\boldsymbol{z}$ corresponding to the collection of indices $\mathcal{V}^*$. Sets $\sum_{i\in\mathcal{V}^*} z_i = \widehat{k}s - z'$ for uniformly chosen $\widehat{k} \in [k']$. Now, we define the function K(gid) as

$$\mathrm{K}(\mathsf{gid}) = \begin{cases} 0, & \text{if } z' + \sum_{i\in\mathcal{V}} z_i \equiv 0 \mod s \\ 1, & \text{elsewhere} \end{cases}$$

So, from the above definition of the function K, we can say that $\mathrm{K}(\mathsf{gid}^*) = 0$ and for all $\mathsf{gid}(\neq \mathsf{gid}^*)$ it becomes non-zero. Additionally, we set another two functions as $\mathrm{F}(\mathsf{gid}) = p - s\widehat{k} + z' + \sum_{i\in\mathcal{V}} z_i$ and $\mathrm{J}(\mathsf{gid}) = w' + \sum_{i\in\mathcal{V}} w_i$. The simulator assigns the public parameters $u' = g_2^{p - \widehat{k}s + z'} \cdot g^{w'}$ and $u_i = g_2^{z_i} \cdot g^{w_i}$. From the adversarial perspective, the distribution of the public parameters is identical to the real construction.

Now, we construct a PPT adversary $\mathscr{B}$ that breaks the DBDH assumption 2 with non-negligible advantage.

We construct a PPT reduction $\mathscr{B}$ which breaks the DBDH assumption 2 with non-negligible advantage. The reduction algorithm $\mathscr{B}$ first receives the DBDH challenge from the challenger as $(\mathbb{BG}, g^a, g^b, g^c, e(g,g)^\tau)$ where $\mathbb{BG} = (q, g, g_2, \mathbb{G}, \mathbb{G}_T, e(\cdot,\cdot))$ is a group description,

$a, b, c \leftarrow \mathbb{Z}_q$ are random integers and the element $e(g,g)^{\tau} \in \mathbb{G}_T$ is either $e(g,g)^{abc}$ or a random group element from the target group $\mathbb{G}_T$. In the following, we discuss about the simulation of the master public key, queried secret keys and the challenge ciphertext. The algorithm $\mathscr{B}$ works as follows:

Public key simulation: The adversary $\mathscr{B}$ implicitly sets the following vectors of length m as

$$\boldsymbol{a} = (a, a_2, \ldots, a_m), \boldsymbol{b} = (b, b, \ldots, b), \boldsymbol{c} = (c, c, \ldots, c)$$

where it randomly samples $a_2, \ldots, a_m \leftarrow \mathbb{Z}_q$. Let us consider the notation $\boldsymbol{u} \odot \boldsymbol{v}$ by component wise multiplication of the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$. In this case, the notation $\boldsymbol{a} \odot \boldsymbol{b} = (ab, a_2 b, \ldots, a_m b) = b\boldsymbol{a}$. To generated the public key, $\mathscr{B}$ implicitly set $\widetilde{\boldsymbol{\alpha}} = \boldsymbol{a}$ and returns the master public key as

$$\mathsf{mpk} = \left(g^{\mathbf{F}^{\top}\boldsymbol{a}}, g_2 = g^b, u', g\right)$$

where the exponent $g^{\boldsymbol{a}}$ is computed as follows:

$$g^{\boldsymbol{a}} = \left(g^a, g^{a_2}, \ldots, g^{a_m}\right) = g^{\boldsymbol{\alpha}}$$

Note that, $a_2, \ldots, a_m$ are distributed uniformly over $\mathbb{Z}_q$ and hence the public key components are properly simulated by using the DBDH instances.

Challenge ciphertext simulation. To generate the challenge ciphertext, $\mathscr{B}$ chooses the random exponent $t \leftarrow \mathbb{Z}_q$ and implicitly sets $c = t$. We now show that how does $\mathscr{B}$ simulate the challenge ciphertext by using the DBDH instances.

$$\begin{aligned}
\boldsymbol{C}_1 &= e(g,g)^{\mathbf{F}^{\top} bc(a,a_2,\ldots,a_m)} \cdot e(g,g)^{\mathbf{F}^{\top}(-b\mathfrak{b}\boldsymbol{e}_1 + b\mathbf{F}\boldsymbol{x}^{(0)})} \\
&= e(g,g_2)^{\mathbf{F}^{\top}(c(a,a_2,\ldots,a_m) - \mathfrak{b}\boldsymbol{e}_1 + \mathbf{F}\boldsymbol{x}^{(0)})} \\
&= e(g,g_2)^{\mathbf{F}^{\top}(c(a,a_2,\ldots,a_m) - \mathfrak{b}\boldsymbol{e}_1 + \mathbf{F}\boldsymbol{x}^{(0)})} \\
&= e(g,g_2)^{\mathbf{F}^{\top}(\widetilde{\boldsymbol{\alpha}} t - \mathfrak{b}\boldsymbol{e}_1 + \mathbf{F}\boldsymbol{x}^{(0)})}
\end{aligned}$$

$$C_2 = g^t = g^c$$

$$C_3 = g^{c\mathsf{J}(\mathsf{gid}^*)} = \mathrm{H}(\mathsf{gid}^*)^c$$

Secret key simulation. $\mathscr{B}$ answers the secret key $\mathsf{sk}_{\boldsymbol{y}}$ associated to an identity gid and a predicate vector $\boldsymbol{y}$ as described below.

We consider two different cases based on queried identity gid. For an identity gid, consider $\mathscr{V} \subseteq \{1, \ldots, k'\}$ be the set of all i for which $\mathsf{gid}_i = 1$.

Case 1: If $\mathsf{gid} \neq \mathsf{gid}^*$, $\mathscr{B}$ simulates the secret key as follows:

By using the technique of Boneh and Boyen [BB04] and Waters IBE [Wat05] scheme, $\mathscr{B}$ randomly chooses $r \leftarrow \mathbb{Z}_p$ then it simulates the secret key $\mathsf{sk}_{\boldsymbol{y}} = (d_1, d_2)$ associated to an identity gid and a vector $\boldsymbol{u}$ as

$$\begin{aligned}
d_1 &= g^{-\frac{\mathsf{J}(\mathsf{gid})}{\mathsf{F}(\mathsf{gid})} \cdot \langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y} \rangle} (u' \prod_{i \in \mathscr{V}} u_i)^r \\
&= g_2^{\langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y} \rangle} (g_2^{\mathsf{F}(\mathsf{gid})} g^{\mathsf{J}(\mathsf{gid})})^{\frac{-\langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y} \rangle}{\mathsf{F}(\mathsf{gid})}} (g_2^{\mathsf{F}(\mathsf{gid})} g^{\mathsf{J}(\mathsf{gid})})^r \\
&= g_2^{\langle \widetilde{\boldsymbol{\alpha}}, \mathbf{F}\boldsymbol{y} \rangle} (g_2^{\mathsf{F}(\mathsf{gid})} g^{\mathsf{J}(\mathsf{gid})})^{r - \frac{\langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y} \rangle}{\mathsf{F}(\mathsf{gid})}} \\
&= g_2^{\langle \widetilde{\boldsymbol{\alpha}}, \mathbf{F}\boldsymbol{y} \rangle} \mathrm{H}(\mathsf{gid})^{r - \frac{\langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y} \rangle}{\mathsf{F}(\mathsf{gid})}} \\
d_2 &= g^r g^{-\langle (a, a_2, \ldots, a_m), \mathbf{F}\boldsymbol{y} \rangle \frac{1}{\mathsf{F}(\mathsf{gid})}} \\
&= g^{r - \langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y} \rangle \frac{1}{\mathsf{F}(\mathsf{gid})}} \\
&= g^{r - \langle \widetilde{\boldsymbol{\alpha}}, \mathbf{F}\boldsymbol{y} \rangle \frac{1}{\mathsf{F}(\mathsf{gid})}}
\end{aligned}$$

We implicitly set $r' = r - \langle \widetilde{\boldsymbol{\alpha}}, \mathbf{F}\boldsymbol{y} \rangle \frac{1}{\mathsf{F}(\mathsf{gid})}$. So, from the construction of K function, we can conclude that $\mathrm{K}(\mathsf{gid}) \neq 0$ for any key query corresponding to the identity gid. This implies that

the function $\mathrm{F}(\mathsf{gid}) \neq 0 \mod N$ for a particular identity (as $p > sn$ for any reasonable values of p, n and s). To prove the above conclusion, we need the following Lemma 24.

Case 2: If $\mathsf{gid} = \mathsf{gid}^*$, in this case, $\mathscr{B}$ responds the secret key $\mathsf{sk}_{\boldsymbol{y}}$ as follows:

$$\begin{aligned}
d_1 &= g^{b\mu}(g_2^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid})^*})^r \\
&= g^{\langle \boldsymbol{a}\odot\boldsymbol{b}, \mathbf{F}\boldsymbol{y}\rangle}(g_2^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r \\
&= g^{\langle b(a,a_2,\ldots,a_m), \mathbf{F}\boldsymbol{y}\rangle}(g_2^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r \\
&= g_2^{\langle (a,a_2,\ldots,a_m), \mathbf{F}\boldsymbol{y}\rangle}(g_2^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r \\
&= g_2^{\langle (a,a_2,\ldots,a_m), \mathbf{F}\boldsymbol{y}\rangle}(g_2^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r \\
&= g_2^{\langle \boldsymbol{a}, \mathbf{F}\boldsymbol{y}\rangle}(g_2^{\mathrm{F}(\mathsf{gid}^*)} g^{\mathrm{J}(\mathsf{gid}^*)})^r \\
&= g_2^{\langle \widetilde{\boldsymbol{\alpha}}, \mathbf{F}\boldsymbol{y}\rangle}\mathrm{H}(\mathsf{gid}^*)^r \\
d_2 &= g^r
\end{aligned}$$

where the second equality follows from the fact that $\langle \boldsymbol{a}\odot\boldsymbol{b}, \mathbf{F}\boldsymbol{y}\rangle = b\mu$ and $\mu \in \mathbb{Z}_q$ is known to the challenger $\mathscr{B}$. From the formation of $\mathbf{F}$, we have the relation $\mathbf{F}(\boldsymbol{x}^{(0)} - \boldsymbol{x}^{(1)}) = (1, 0, \ldots, 0)^\top = \boldsymbol{e}_1$. Since in this case, $\mathsf{gid} = \mathsf{gid}^*$ so, the queried secret key vector $\boldsymbol{y}$ should satisfy the relation $\langle \boldsymbol{x}^{(0)} - \boldsymbol{x}^{(1)}, \boldsymbol{y}\rangle = 0$. Therefore, we have $\langle \boldsymbol{e}_1, \mathbf{F}\boldsymbol{y}\rangle = 0$ which implies that $\langle (\boldsymbol{a}\odot\boldsymbol{b}), \mathbf{F}\boldsymbol{y}\rangle = \langle (ab, a_2 b, \ldots, a_m b), \mathbf{F}\boldsymbol{y}\rangle = b\mu$ for any $\mu \in \mathbb{Z}_q$.

Guess. If $\mathscr{A}$ guesses the challenge bit $\mathfrak{b} \leftarrow \{0,1\}$ correctly then $\mathscr{B}$ returns 1 otherwise it outputs 0. We consider $\boldsymbol{w} = bc(a, a_2, \ldots, a_m) = (\tau, bca_2, \ldots, bca_m)$ where $e(g,g)^\tau$ is the challenge element. If $\tau = abc$, then all the secret keys and challenge ciphertext are properly distributed. In particular, the challenge ciphertext is an encryption of the message vector $\boldsymbol{x}^{(\mathfrak{b})}$. Therefore, in this case, $\mathscr{A}$ outputs $\mathfrak{b}' = \mathfrak{b}$ with advantages $\frac{1}{2} + \mathsf{negl}(\lambda)$ where $\mathsf{negl}(\lambda)$ is the advantage of $\mathscr{A}$ in the selective security game of the IBIPFE. Otherwise, if τ is randomly generated from $\mathbb{Z}_q$ then the challenge ciphertext component $\boldsymbol{C_1}$ uniform element from the target group $\mathbb{G}_T$. So, $\mathscr{A}$ can not get any informations about the challenge bit $\mathfrak{b}$ from this component. So, $\mathscr{A}$ wins the game with the probability $\frac{1}{2}$. Hence, from the hardness of DBDH assumption, it can conclude that $\mathscr{A}$ has a non-negligible advantages against the proposed IBIPFE scheme achieves the selective security. This completes the proof.

Lemma 24 *For any Q-secret key query corresponding to the identities $\mathsf{gid}^{(1)}, \mathsf{gid}^{(2)}, \ldots, \mathsf{gid}^{(Q)}$ to the key generation oracle, the probabilities of $\mathrm{K}(\mathsf{gid}^{(\ell)}) = 1$ with $z' + \sum_{i\in\mathcal{V}^*} z_i = \widehat{k}s$ is non-negligible for all ℓ.*

Proof. For any set of Q-queries corresponding to the identities $\mathsf{gid}^{(1)}, \mathsf{gid}^{(2)}, \ldots, \mathsf{gid}^{(Q)}$ and the challenge identity gid^*, we compute the following probability.

$$\begin{aligned}
&\Pr\left[\bigwedge_{\ell=1}^{Q}(\mathrm{K}(\mathsf{gid}^{(\ell)}) = 1) | (z' + \sum_{i\in\mathcal{V}^*} z_i) = \widehat{k}s\right] \\
&= \Pr\left[\bigwedge_{\ell=1}^{Q}(\mathrm{K}(\mathsf{gid}^{(\ell)}) = 1 | \mathrm{K}(\mathsf{gid}^*) = 0)\right] \\
&= \left(1 - \Pr\left[\bigvee_{\ell=1}^{Q}(\mathrm{K}(\mathsf{gid}^{(\ell)}) = 0) | \mathrm{K}(\mathsf{gid}^*) = 0)\right]\right)
\end{aligned}$$

$$\geq \left(1 - \sum_{\ell=1}^{Q} \Pr\left[(\mathrm{K}(\mathrm{gid}^{(\ell)}) = 0) | \mathrm{K}(\mathrm{gid}^{*}) = 0)\right]\right)$$
$$= \left(1 - \frac{Q}{s}\right) = 0.9$$

We can optimize the last equation by setting $s = 10Q$ (as we did in the simulation), where Q is the maximum number of queries. This above result shows that for all queried identities in the key generation oracle except the challenge identity gid, the K values should be equals to 1 with overwhelming probability.