SCIENCE CHINA
Information Sciences

• **Supplementary File** •

Simulation-based security of function-hiding inner product encryption

Qingsong ZHAO[1,2,4], Qingkai ZENG[1,2*], Ximeng LIU[3] & Huanliang XU[4]

[1]*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210023*, China;*
[2]*Department of Computer Science and Technology, Nanjing University, Nanjing* 210023*, China;*
[3]*School of Information Systems, Singapore Management University, Singapore 178902, Singapore;*
[4]*College of Information Science and Technology, Nanjing Agricultural University, Nanjing* 210095*, China*

Appendix A Simulation-based security

Definition 1 (Simulation-based security)**.** An IPE scheme has (adaptive) simulation-based security if for all PPT adversaries $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, there exits a PPT simulator $S = (S_1, S_2)$ such that it is computationally indistinguishable between the following two distributions.

$\mathbf{Real}_{\mathcal{A}}^{\mathrm{IPE}}(1^\lambda)$:	$\mathbf{IDEAL}_{\mathcal{A}}^{\mathrm{IPE}}(1^\lambda)$:
(MSK, PP)$\xleftarrow{\$}$ Setup$(1^\lambda, n)$	(MSK, PP)$\xleftarrow{\$}$ Setup$(1^\lambda, n)$
$(\vec{x}, st) \xleftarrow{\$} \mathcal{A}_1^{\mathrm{KeyGen(MSK},\cdot)}$(PP)	$(\vec{x}, st) \xleftarrow{\$} \mathcal{A}_1^{\mathrm{KeyGen(MSK},\cdot)}$(PP)
CT $\xleftarrow{\$}$ Encrypt(MSK, PP, $\vec{x}$)	(CT, $st^{'}$) $\xleftarrow{\$}$ S_1(PP, $1^{\lvert\vec{x}\rvert}$, $\{\mathrm{SK}_i, \langle x^{(j)}, y^{(i)}\rangle\}$)
$\alpha \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}(\mathrm{MSK},\cdot)}$(PP, CT, st)	$\alpha \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}^{'}(\mathrm{MSK},st^{'},\cdot)}$(PP, CT, st)
output($\alpha, \vec{x}$, PP)	output(α, m, PP)

The security notion requires that any message that adversaries receive from ciphertexts and secret keys, can be simulated by a simulator given only oracle $\mathcal{O}(\mathrm{MSK}, \cdot)$ and oracle $\mathcal{O}^{'}(\cdot)$ to which adversaries submit each pair of vectors. The oracle $\mathcal{O}(\mathrm{MSK}, \cdot)$ is equal to KeyGen(MSK, $\cdot$), and the oracle $\mathcal{O}^{'}(\cdot)$ is the second stage of the simulator, i.e., $S_2^{\langle x^{(j)}, y^{(i)}\rangle}(\mathrm{MSK}, \mathrm{st}^{'}, \cdot)$, where $x^{(j)} \in \mathbb{Z}_q^n \setminus \{\vec{0}\}$ and $y^{(i)} \in \mathbb{Z}_q^n \setminus \{\vec{0}\}$ are inputs of the j^{th} ciphertext query and the i^{th} secret key query by adversaries respectively. In addition, the simulator S_2 is stateful and after each query it updates the $st^{'}$ (the simulator S_1 initializes the state) which is transfered to its next secret key query. Besides that, key queries in the first stage are generated honestly (not by the simulator) and Setup is replaced by $\widetilde{\mathbf{Setup}}$ in the ideal experiment, which are in accordance with [1]. This means the simulator generates the public parameters of the ideal experiment.

Remark 1. The notion of simulation-based security is stronger than the indistinguishability-based security definition, so an IPE scheme that is simulation-based secure also has indistinguishability-based security [2].

Remark 2. Simulation-based security is impossible against the adversary that sees an unbounded number of ciphertexts in the standard model [2–4], because the size of the ciphertexts has to grow along with the number of (adaptive) key queries of the adversary. However, the lower bound do not work when there is a bound number of challenge ciphertexts or the output of **Decrypt** is in a polynomial-size range. Moreover, the lower bound does not apply when a FE scheme is constructed in the generic group model [5] or the random oracle model [6, 7]. The random oracle model could be leveraged to bypass the impossibility of simulation-based security by allowing the simulator to make the queries that the adversary makes to the random oracle in the real experiment. Boneh et al. [2] showed that public index schemes are impossible to simulate in the standard model, but possible in the random oracle model. It raises an interesting issue how to extend inner product encryption with either an impossibility or a possibility for an unbounded number of ciphertext queries.

* Corresponding author (email: zqk@nju.edu.cn)

Table B1 Complexity comparison of IPE schemes in the private-key setting. n is the dimension of inner product vectors, $|\mathbb{Z}_q|$ is the length of a member in the finite field $\mathbb{Z}_q$, $|\mathbb{G}_i|$ is the length of an element in group $\mathbb{G}_i$ for each $i \in \{0,1\}$.

	BJK15 [8]	DDM16 [9]	SIPE scheme
♯ Master Secret Key	$(8n^2+8)\|\mathbb{Z}_q\|$	$(8n^2+12n+28)\|\mathbb{Z}_q\|$	$(6n^2+10n+24)\|\mathbb{Z}_q\|$
♯ Ciphertext	$(2n+2)\|\mathbb{G}_1\|$	$(4n+8)\|\mathbb{G}_1\|$	$(2n+4)\|\mathbb{G}_1\|$
♯ Secret Key	$(2n+2)\|\mathbb{G}_2\|$	$(4n+8)\|\mathbb{G}_2\|$	$(n+3)\|\mathbb{G}_2\|$
♯ Scalar Multiplications in **Encrypt**	$2n+2$	$4n+8$	$2n+4$
♯ Scalar Multiplications in **KeyGen**	$2n+2$	$4n+8$	$n+3$
♯ Pairing Operations	$2n+2$	$4n+8$	$2n+4$

Appendix B Comparison with previous schemes

Efficiency. In terms of storage complexity, the ciphertexts and the secret keys of the SIPE scheme are comprised of $2n+4$ and $n+3$ group elements respectively whereas the master secret key achieves $6n^2+10n+24$ members of the finite field $\mathbb{Z}_q$. Regarding computation complexity, the number of scalar multiplications on cyclic groups is $2n+4$ in the key generation algorithm and $n+3$ in the encryption algorithm respectively while the decryption algorithm uses $2n+4$ pairing operations. Notice that performance in the SIPE scheme is better than that in [8, 9] in both storage and computation efficiency. We provide a comparison in Table B1.

Discussion. In the construction of the SIPE scheme, we introduce two pairs of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$, which have $2n+2$ dimensions and 4 dimensions respectively, where n is the dimension of inner product vectors. The $2n+2$ dimension $\mathbb{B}$ and the 4 dimension $\mathbb{D}$ are used to encode a vector $\vec{x}$ while $n+2$ and 3 dimensions of $\mathbb{B}^*$ and $\mathbb{D}^*$ respectively are used to encode a vector $\vec{y}$. We preserve the remaining hidden dimensions of $\mathbb{B}^*$ and $\mathbb{D}^*$ for the security reduction. In [9], two pairs of dual orthonormal bases are also used, which have $4n+2$ dimensions and 6 dimensions respectively. The $n+2$ dimensions of the first pair of bases and 3 dimensions of the second pair are used to encode ciphertext vectors and secret key vectors respectively while the remaining dimensions are preserved to realize the various forms of ciphertexts and secret keys in the security reduction. Although the construction of [8] also employs two pairs of dual orthonormal bases, which are $2n$-dimensional and 2-dimensional respectively, they are entirely used to construct the IPE scheme and ciphertext vectors and secret key vectors are encoded twice to create space for the security reduction respectively.

For the security proof, we provide a simulation-based proof [1, 10] while [8, 9] adopt an indistinguishability-based proof where an adversary cannot distinguish if a challenger is asking to compute a ciphertext on input x_0 and x_1. Simulation-based proofs construct a simulator and have to prove the simulator's view in an execution of a scheme is indistinguishable from the real execution.

A key point in our construction is that there is asymmetric structure between secret keys and ciphertexts. Typically, a ciphertext associated with a vector $\vec{x}$ employs two parallel systems of $\vec{x}$ in the exponent of c_1: one is the coefficient of $\vec{b}_i$ and the other is the coefficient of $\vec{b}_{n+i}$ for $i = 1, ..., n$. Different from this, a secret key for a vector $\vec{y}$ employs a copy of $\vec{y}$ in the exponent of k_1^*. This allows us to use the features of hidden linear subspaces of DPVS information-theoretically in prime order bilinear group setting. Specially, a hidden subspace can be applied to indicate that a ciphertext vector $\vec{x}_1$ can be converted to another ciphertext vector $\vec{x}_2$ having the same inner product with a underlying key vector $\vec{y}$. Also, the asymmetric structure is inherent in our construction. It is because a simulated secret key will correctly decrypt a simulated ciphertext, while a simulated secret key will incorrectly decrypt a normal ciphertext in a hybrid argument over the simulated views. It is contrast to IPE schemes, where the structure of secret keys and ciphertexts is in a symmetric fashion [8, 9], achieving indistinguishability-based security by methodology of the dual system encryption [11]. In the dual system encryption, a semi-functional secret key can decrypt a normal ciphertext and a normal secret key also can decrypt a semi-functional ciphertext, but a semi-functional secret key can not decrypt a semi-functional ciphertext. Although there is some resemblance between our construction and that of [8, 9], they can only be proved to be indistinguishability-based secure due to implicitly using the dual system encryption technique.

Appendix C Security proof

In this section, we prove our SIPE scheme is simulation-based secure from the SXDH assumption.

Theorem 1. Under the SXDH assumption the inner product encryption scheme is one-AD-SIM-secure.

Proof. The proof of theorem 1 employs the following result.

Lemma 1 (Lemma 3 in [12]). For $\varsigma \in \mathbb{Z}_q$, Let $\mathbb{C}_\varsigma = \{(\vec{x}, \vec{y}) | \langle \vec{x}, \vec{y} \rangle = \varsigma\} \subset \mathbb{Z}_q^n \times \mathbb{Z}_q^n$, where n denotes a positive integer and q denotes a prime integer. For all $(\vec{x}, \vec{y}) \in \mathbb{C}_\varsigma$, for all $(\vec{t}, \vec{w}) \in \mathbb{C}_\varsigma$,

$$\Pr[\vec{x}\mathbf{M} = \vec{t} \wedge \vec{y}\mathbf{M}' = \vec{w}] = 1/\sharp\mathbb{C}_\varsigma,$$

Figure C1 Experiments for the proof of Theorem 1.

$Real^{\mathcal{A}}$:	$H_1^{\mathcal{A}}$:
Setup$(1^\lambda, \mathrm{n})\rightarrow$ MSK, PP	$\widetilde{\textbf{Setup}}(1^\lambda, \mathrm{n})\rightarrow \widetilde{\mathrm{MSK}}, \widetilde{\mathrm{PP}}$
Encrypt(MSK, PP, $\vec{x}$)$\rightarrow$ CT	**Encrypt**(MSK, PP, $\vec{x}$)$\rightarrow$ CT
KeyGen(MSK, PP, $\vec{y}$)$\rightarrow$ SK	**KeyGen**(MSK, PP, $\vec{y}$)$\rightarrow$ SK
$H_2^{\mathcal{A}}$:	$Ideal^{\mathcal{A}}$:
$\widetilde{\textbf{Setup}}(1^\lambda, \mathrm{n})\rightarrow \widetilde{\mathrm{MSK}}, \widetilde{\mathrm{PP}}$	$\widetilde{\textbf{Setup}}(1^\lambda, \mathrm{n})\rightarrow \widetilde{\mathrm{MSK}}, \widetilde{\mathrm{PP}}$
$\widetilde{\textbf{Encrypt}}$(MSK, PP, $\vec{x}$)$\rightarrow \widetilde{\mathrm{CT}}$	$\widetilde{\textbf{Encrypt}}$(MSK, PP, $\vec{x}$)$\rightarrow \widetilde{\mathrm{CT}}$
KeyGen(MSK, PP, $\vec{y}$)$\rightarrow$ SK	$\widetilde{\textbf{KeyGen}}$(MSK, PP, $\vec{y}$)$\rightarrow \widetilde{\mathrm{SK}}$

where $\mathbf{M} \xleftarrow{\$} GL(n, \mathbb{Z}_q)$, $\mathbf{M}' = (\mathbf{M}^{-1})^\top$, and the cardinality of a set C is denoted by $\sharp\mathbb{C}$.

To prove the theorem, we use a simulator Sim={$\widetilde{\textbf{Setup}}$, $\widetilde{\textbf{Encrypt}}$, $\widetilde{\textbf{KeyGen}}$}.

$\widetilde{\textbf{Setup}}$: produces public parameters $\widetilde{\mathrm{PP}}$, which is delivered to the adversary, and a master secret key $\widetilde{\mathrm{MSK}}$. The pubic parameters and the master secret key are leveraged to answer the queries of the adversary. In more detail, the algorithm samples dual orthonormal bases $(\mathbb{F}, \mathbb{F}^*) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $(\mathbb{H}, \mathbb{H}^*) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$, defines $(\mathbb{B}, \mathbb{B}^*)$ with $(\mathbb{F}, \mathbb{F}^*)$, and sets $(\mathbb{D}, \mathbb{D}^*)$ with $(\mathbb{H}, \mathbb{H}^*)$. In the master secret key $\widetilde{\mathrm{MSK}}$, $(\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$ is consistent with $(\mathbb{B}, \mathbb{B}^*, \mathbb{D}, \mathbb{D}^*)$. Notice that $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ have the same distribution of those output of $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$ respectively.

$\widetilde{\textbf{Encrypt}}$: simulates the ciphertext of the challenge plaintext $\vec{x}$. It takes in $\widetilde{\mathrm{MSK}}$, $\widetilde{\mathrm{PP}}$, and the values $(\vec{y}_i, t_i)_{i=1}^{p_1}$, where $t_i = \langle \vec{x}, \vec{y}_i \rangle$ and p_1 is the number of key queries in the first stage. The normal form of the ciphertext is (c_1, c_2) with $c_1 = g_1^{\alpha \sum_{i=1}^n x_i \vec{b}_i + \alpha' \sum_{i=1}^n x_i \vec{b}_{n+i} + \varphi \vec{b}_{2n+2}}$ (thereafter c_2 is ignored since c_2 is only used to remove same scalars in the exponent from c_1 and so is k_2^* in (k_1^*, k_2^*)). The simulated ciphertext is $c_1 = g_1^{\alpha \sum_{i=1}^n x_i \vec{b}_i + \alpha' \sum_{i=1}^n t_i \vec{b}_{n+i} + \varphi \vec{b}_{2n+2}}$, where $\vec{t} = \vec{x}\mathbf{M}$ and $\mathbf{M}$ is a random regular $(n \times n)$-matrix. This change can be done since zero vector 0^n is used for the correspond part in the normal form of the secret key, and the SIPE scheme for inner product is in the private-key setting.

$\widetilde{\textbf{KeyGen}}$: simulates the answers corresponding to the second stage queries of the adversary. It takes in $\widetilde{\mathrm{MSK}}$, $\widetilde{\mathrm{PP}}$, the vector $\vec{y}$ for which the simulator must simulate the secret key, and the value $t = \langle \vec{x}, \vec{y} \rangle$, where $\vec{x}$ is the challenge message. The normal form of the secret key is $k_1^* = g_2^{\beta \sum_{i=1}^n y_i \vec{b}_i^* + \eta \vec{b}_{2n+1}^*}$. The simulated secret key is $k_1^* = g_2^{\beta \sum_{i=1}^n y_i \vec{b}_i^* + \beta' \sum_{i=1}^n w_i \vec{b}_{n+i}^* + \eta \vec{b}_{2n+1}^*}$, where $\vec{w} = \vec{y}(\mathbf{M}^\top)^{-1}$ and $\beta' \xleftarrow{\$} \mathbb{Z}_q$, and it can not decrypt a normal ciphertext. However, it can correctly decrypt a simulated ciphertext, since $\langle \vec{t}, \vec{w} \rangle = (\vec{x}\mathbf{M}) \cdot (\vec{y}(\mathbf{M}^\top)^{-1}) = \langle \vec{x}, \vec{y} \rangle$, i.e., $E(c_1, k_1^*) = e(g_1, g_2)^{(\alpha\beta + \alpha'\beta')\langle \vec{x}, \vec{y} \rangle}$. Note that $\langle \vec{t}, \vec{w} \rangle$ are uniformly and pairwise-independently distributed in $\mathbb{C}_{\varsigma_{t,w}}$ from Lemma 1, where $\langle \vec{t}, \vec{w} \rangle = \varsigma_{t,w}$. Therefore, the joint distribution of the simulated ciphertext and the simulated secret key, except with the probability $1/\sharp\mathbb{C}_\varsigma$, has the equivalence of that of an independent pair of the normal ciphertext and the normal secret key.

We proceed via a series of experiments described in Figure C1. To prove the theorem 1, it requires that the output of the experiment $Real^{\mathcal{A}}$ is computationally indistinguishable from that of the experiment $Ideal^{\mathcal{A}}$ for all PPT adversaries $\mathcal{A}$. The formal proof follows a rather standard hybrid argument. Thus, we define two intermediate hybrids $H_1^{\mathcal{A}}, H_2^{\mathcal{A}}$. $H_1^{\mathcal{A}}$ is the same as $Real^{\mathcal{A}}$ except that $\widetilde{\textbf{Setup}}$ is used in $H_1^{\mathcal{A}}$ instead of the algorithm **Setup** in $Real^{\mathcal{A}}$. $H_2^{\mathcal{A}}$ is the same as $H_1^{\mathcal{A}}$ except that $\widetilde{\textbf{Encrypt}}$ is used in $H_2^{\mathcal{A}}$ instead of the algorithm **Encrypt** in $H_1^{\mathcal{A}}$. $H_2^{\mathcal{A}}$ is the same as $Ideal^{\mathcal{A}}$ except that $\widetilde{\textbf{KeyGen}}$ is used in $Ideal^{\mathcal{A}}$ instead of the algorithm **KeyGen** in $H_2^{\mathcal{A}}$. We show that no PTT adversary can distinguishable the two. First we give the proof for transition from $Real^{\mathcal{A}}$ to $H_1^{\mathcal{A}}$. We then give the proof for transition from $H_1^{\mathcal{A}}$ to $H_2^{\mathcal{A}}$, to $Ideal^{\mathcal{A}}$.

We prove the equivalence of the distribution of the view of $\mathcal{A}$ in $Real^{\mathcal{A}}$ and that in $H_1^{\mathcal{A}}$ in Lemma 2.

Lemma 2. For all PPT adversaries $\mathcal{A}$, $Real^{\mathcal{A}} \approx_c H_1^{\mathcal{A}}$.

Proof. We construct a PPT algorithm $\mathcal{B}$ that simulates $Real^{\mathcal{A}}$ or $H_1^{\mathcal{A}}$. Then, we observe the distribution of the view of an adversary $\mathcal{A}$ in $H_1^{\mathcal{A}}$ as well as $Real^{\mathcal{A}}$.

Setup: $\mathcal{B}$ runs $\mathcal{G}_{abpg}(1^\lambda)$ to generate an asymmetric bilinear group $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. It then samples dual pairing vector spaces $(q, \mathbb{V}_1, \mathbb{V}_1^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, E_1) \xleftarrow{\$} \mathcal{G}_{dpvs}(1^\lambda, 2n+2, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$ and $(q, \mathbb{V}_2, \mathbb{V}_2^*, \mathbb{G}_T, \mathbb{A}_2, \mathbb{A}_2^*, E_2) \xleftarrow{\$} \mathcal{G}_{dpvs}(1^\lambda, 4, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$. Next, it samples dual orthonormal bases $(\mathbb{F} = \{\vec{f}_1, ..., \vec{f}_{2n+2}\}, \mathbb{F}^* = \{\vec{f}_1, ..., \vec{f}_{2n+2}^*\}) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $(\mathbb{H} = \{\vec{h}_1, ..., \vec{h}_4\}, \mathbb{H}^* = \{\vec{h}_1^*, ..., \vec{h}_4^*\}) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$. It chooses $\mathbf{M} \xleftarrow{\$} GL(n, \mathbb{Z}_q)$ and sets

$$\begin{aligned}
&\vec{b}_i = \vec{f}_i (i=1,...,n), && \vec{b}_i^* = \vec{f}_i^* + \mu \vec{f}_{n+i}^* (i=1,...,n),\\
&\vec{b}_{2n+i} = \vec{f}_{2n+i} (i=1,2), && \vec{b}_{2n+i}^* = \vec{f}_{2n+i}^* (i=1,2),\\
&\vec{d}_i = \vec{h}_i (i=1,3,4), && \vec{d}_i^* = \vec{h}_i^* (i=2,3,4),\\
&\vec{d}_2 = \vec{h}_2 - \mu \vec{h}_1, && \vec{d}_1^* = \vec{h}_1^* + \mu \vec{h}_2^*,
\end{aligned}$$

$$\begin{pmatrix} \vec{b}_{n+1} \\ \vdots \\ \vec{b}_{2n} \end{pmatrix} = \mathbf{M}^{-1} \cdot \begin{pmatrix} \vec{f}_{n+1} - \mu \vec{f}_1 \\ \vdots \\ \vec{f}_{2n} - \mu \vec{f}_n \end{pmatrix}, \begin{pmatrix} \vec{b}_{n+1}^* \\ \vdots \\ \vec{b}_{2n}^* \end{pmatrix} = \mathbf{M}^{\top} \cdot \begin{pmatrix} \vec{f}_{n+1}^* \\ \vdots \\ \vec{f}_{2n}^* \end{pmatrix}.$$

Note that here we abuse the notation for matrix multiplication slightly. We actually intend to define $\vec{b}_{n+i} = \sum_{j=1}^{n} (\mathbf{M}^{-1})_{i,j} (\vec{f}_{n+j} - \mu \vec{f}_j)$, where $(\mathbf{M}^{-1})_{i,j}$ is the i,jth element of $\mathbf{M}^{-1}$, and similarly for $\vec{b}_{n+i}^*$.

Then it implicitly defines $\mathbb{B} = \{\vec{b}_1, ..., \vec{b}_{2n+2}\}$, $\mathbb{B}^* = \{\vec{b}_1^*, ..., \vec{b}_{2n+2}^*\}$, $\mathbb{D} = \{\vec{d}_1, ..., \vec{d}_4\}$, $\mathbb{D}^* = \{\vec{d}_1^*, ..., \vec{d}_4^*\}$, and sets $\widehat{\mathbb{B}} = \{\vec{b}_1, ..., \vec{b}_{2n}, \vec{b}_{2n+2}\}$, $\widehat{\mathbb{B}}^* = \{\vec{b}_1^*, ..., \vec{b}_n^*, \vec{b}_{2n+1}^*\}$, $\widehat{\mathbb{D}} = \{\vec{d}_1, \vec{d}_2, \vec{d}_4\}$, $\widehat{\mathbb{D}}^* = \{\vec{d}_1^*, \vec{d}_3^*\}$. Notice that $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ have the same distribution of those output of $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$ respectively. The reason is that these bases are created by applying an invertible linear transformation to the outputs of $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$. The orthonormality property of $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ follows directly from the orthonormality of $(\mathbb{F}, \mathbb{F}^*)$ and $(\mathbb{H}, \mathbb{H}^*)$ respectively. In more detail, The (i,j) minor of M, denoted by $\triangle_{i,j}$, is obtained by deleting the i-th row and the j-th column from $\mathbf{M}$, where $\mathbf{M} = (m_{i,j})$ is a $n \times n$ matrix for $n \geqslant 2$. Cofactors $\widetilde{m}_{i,j}$ are defined by $(-1)^{i+j} \triangle_{i,j}$. $det\mathbf{M} = \sum_{j=1}^{n} m_{i,j} \widetilde{m}_{i,j}$ denotes the determinant of $\mathbf{M}$. When $det\mathbf{M} \neq 0$, we have

$$\mathbf{M}^{-1} = \frac{1}{det\mathbf{M}} \begin{pmatrix} \widetilde{m}_{1,1} & \dots & \widetilde{m}_{n,1} \\ \vdots & & \vdots \\ \widetilde{m}_{1,n} & \dots & \widetilde{m}_{n,n} \end{pmatrix}.$$

That is, we show

$$\begin{aligned}
\langle \vec{b}_{n+i}, \vec{b}_i^* \rangle &= (\sum_{j=1}^{n} (\mathbf{M}^{-1})_{i,j} (\vec{f}_{n+j} - \mu \vec{f}_j)) \cdot (\vec{f}_i^* + \mu \vec{f}_{n+i}^*)\\
&= (\mathbf{M}^{-1})_{i,1} (\vec{f}_{n+1} - \mu \vec{f}_1) \cdot (\vec{f}_i^* + \mu \vec{f}_{n+i}^*) + ... + (\mathbf{M}^{-1})_{i,n} (\vec{f}_{2n} - \mu \vec{f}_n) \cdot (\vec{f}_i^* + \mu \vec{f}_{n+i}^*)\\
&= (\mathbf{M}^{-1})_{i,i} (\vec{f}_{n+i} - \mu \vec{f}_i) \cdot (\vec{f}_i^* + \mu \vec{f}_{n+i}^*)\\
&= (\mathbf{M}^{-1})_{i,i} (0 + \mu - \mu - 0)\\
&= 0,\\
\langle \vec{b}_{n+i}, \vec{b}_{n+i}^* \rangle &= (\sum_{j=1}^{n} (\mathbf{M}^{-1})_{i,j} (\vec{f}_{n+j} - \mu \vec{f}_j)) \cdot (\sum_{j=1}^{n} (\mathbf{M}^{\top})_{i,j} \vec{f}_{n+j}^*)\\
&= ((\mathbf{M}^{-1})_{i,1} (\vec{f}_{n+1} - \mu \vec{f}_1) + ... + (\mathbf{M}^{-1})_{i,n} (\vec{f}_{2n} - \mu \vec{f}_n)) \cdot ((\mathbf{M}^{\top})_{i,1} \vec{f}_{n+1}^* + ... + (\mathbf{M}^{\top})_{i,n} \vec{f}_{2n}^*))\\
&= (\mathbf{M}^{-1})_{i,1} (\mathbf{M}^{\top})_{i,1} \vec{f}_{n+1} \cdot \vec{f}_{n+1}^* + ... + (\mathbf{M}^{-1})_{i,n} (\mathbf{M}^{\top})_{i,n} \vec{f}_{2n} \cdot \vec{f}_{2n}^*\\
&= ((\mathbf{M}^{-1})_{i,1}, ..., (\mathbf{M}^{-1})_{i,n}) ((\mathbf{M}^{\top})_{i,1}, ..., (\mathbf{M}^{\top})_{i,n})\\
&= \frac{1}{det\mathbf{M}} (\widetilde{m}_{1,i}, ..., \widetilde{m}_{n,i}) (m_{1,i}, ..., m_{n,i})\\
&= 1,\\
\langle \vec{b}_i, \vec{b}_i^* \rangle &= \vec{f}_i \cdot (\vec{f}_i^* + \mu \vec{f}_{n+i}^*)\\
&= 1,\\
\langle \vec{b}_i, \vec{b}_{n+i}^* \rangle &= \vec{f}_i \cdot (\sum_{j=1}^{n} (\mathbf{M}^{\top})_{i,j} \vec{f}_{n+j}^*)\\
&= (\mathbf{M}^{\top})_{i,1} \vec{f}_i \cdot \vec{f}_{n+1}^* + ... + (\mathbf{M}^{\top})_{i,n} \vec{f}_i^* \cdot \vec{f}_{2n}\\
&= 0
\end{aligned}$$

for $i = 1, ..., n$. Finally, $\mathcal{B}$ gives $\mathcal{A}$ the public parameters $\widetilde{\mathrm{PP}} = (q, \mathbb{V}_1, \mathbb{V}_1^*, \mathbb{V}_2, \mathbb{V}_2^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, \mathbb{A}_2, \mathbb{A}_2^*), E_1, E_2)$. The master secret key, $\widetilde{\mathrm{MSK}} = (\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$, is known to $\mathcal{B}$.

Key queries: To answer key queries made by $\mathcal{A}$, $\mathcal{B}$ chooses random $\beta, \eta, \eta' \xleftarrow{\$} \mathbb{Z}_q$ and responds as follows:

$$\begin{aligned}
k_1^* &= g_2^{\beta \sum_{i=1}^{n} y_i \vec{f}_i^* + \eta \vec{f}_{2n+1}^*} (g_2^{\mu})^{\beta \sum_{i=1}^{n} y_i \vec{f}_{n+i}^*}\\
&= g_2^{\beta \sum_{i=1}^{n} y_i (\vec{f}_i^* + \mu \vec{f}_{n+i}^*) + \eta \vec{f}_{2n+1}^*}
\end{aligned}$$

$$
\begin{aligned}
&= g_2^{\beta \sum_{i=1}^{n} y_i \vec{b}_i^* + \eta \vec{b}_{2n+1}^*}, \\
k_2^* &= g_2^{\beta \vec{h}_1^* + \eta' \vec{h}_3^*} (g_2^{\mu})^{\beta \vec{h}_2^*} \\
&= g_2^{\beta(\vec{h}_1^* + \mu \vec{h}_2^*) + \eta' \vec{h}_3^*} \\
&= g_2^{\beta \vec{d}_1^* + \eta' \vec{d}_3^*}.
\end{aligned}
$$

Ciphertext query: To answer ciphertext query that $\mathcal{A}$ makes, $\mathcal{B}$ chooses random $\widehat{\alpha}, \alpha'', \varphi, \varphi' \xleftarrow{\$} \mathbb{Z}_q$, implicitly sets $\alpha = \widehat{\alpha} + \alpha' \mu$ and responds with

$$
\begin{aligned}
c_1 &= g_1^{\widehat{\alpha} \sum_{i=1}^{n} x_i \vec{f}_i + \alpha' \sum_{i=1}^{n} x_i \vec{f}_{n+i} + \varphi \vec{f}_{2n+2}} \\
&= g_1^{(\widehat{\alpha} + \alpha' \mu) \sum_{i=1}^{n} x_i \vec{f}_i + \alpha' \sum_{i=1}^{n} x_i (\vec{f}_{n+i} - \mu \vec{f}_i) + \varphi \vec{f}_{2n+2}} \\
&= g_1^{\alpha \sum_{i=1}^{n} x_i \vec{b}_i + \alpha' \sum_{i=1}^{n} x_i \vec{b}_{n+i} + \varphi \vec{b}_{2n+2}}, \\
c_2 &= g_1^{\widehat{\alpha} \vec{h}_1 + \alpha' \vec{h}_2 + \varphi' \vec{h}_4} \\
&= g_1^{(\widehat{\alpha} + \alpha' \mu) \vec{h}_1 + \alpha' (\vec{h}_2 - \mu \vec{h}_1) + \varphi' \vec{h}_4} \\
&= g_1^{\alpha \vec{d}_1 + \alpha' \vec{d}_2 + \varphi' \vec{d}_4}.
\end{aligned}
$$

The public parameters PP, the answer of the queries (k_1^*, k_2^*) and the ciphertext query (c_1, c_2) constitute the view of the adversary $\mathcal{A}$. Note that $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{D}, \mathbb{D}^*)$ have the same distribution of those output of $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $\mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$ respectively. The answers of the key queries are distributed as in $Real^{\mathcal{A}}$ and as in $H_1^{\mathcal{A}}$ which is similar to the answer to the ciphertext query. Thus the view of $\mathcal{A}$ in $Real^{\mathcal{A}}$ and that in $H_1^{\mathcal{A}}$ have the same distribution. □

We prove the equivalence of the distribution of the view of $\mathcal{A}$ in $H_1^{\mathcal{A}}$ and that in $H_2^{\mathcal{A}}$ in Lemma 3.

Lemma 3. For all PPT adversaries $\mathcal{A}$, $H_1^{\mathcal{A}} \approx_c H_2^{\mathcal{A}}$.

Proof. We construct a PPT algorithm $\mathcal{B}$ that simulates $H_1^{\mathcal{A}}$ or $H_2^{\mathcal{A}}$. Then, we observe the distribution of the view of an adversary $\mathcal{A}$ in $H_1^{\mathcal{A}}$ as well as $H_2^{\mathcal{A}}$.

Setup: $\mathcal{B}$ samples dual orthonormal bases $(\mathbb{F} = \{\vec{f}_1, ..., \vec{f}_{2n+2}\}, \mathbb{F}^* = \{\vec{f}_1, ..., \vec{f}_{2n+2}^*\}) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^{2n+2})$ and $(\mathbb{H} = \{\vec{h}_1, ..., \vec{h}_4\}, \mathbb{H}^* = \{\vec{h}_1^*, ..., \vec{h}_4^*\}) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, \mathbb{Z}_q^4)$. It chooses $\mathbf{M} \xleftarrow{\$} GL(n, \mathbb{Z}_q)$ and sets

$$
\begin{aligned}
&\vec{b}_i = \vec{f}_i (i = 1, ..., n), && \vec{b}_i^* = \vec{f}_i^* + \mu \vec{f}_{n+i}^* (i = 1, ..., n), \\
&\vec{b}_{2n+i} = \vec{f}_{2n+i} (i = 1, 2), && \vec{b}_{2n+i}^* = \vec{f}_{2n+i}^* (i = 1, 2), \\
&\vec{d}_i = \vec{h}_i (i = 1, 3, 4), && \vec{d}_i^* = \vec{h}_i^* (i = 2, 3, 4), \\
&\vec{d}_2 = \vec{h}_2 - \mu \vec{h}_1, && \vec{d}_1^* = \vec{h}_1^* + \mu \vec{h}_2^*,
\end{aligned}
$$

$$
\begin{pmatrix} \vec{b}_{n+1} \\ \vdots \\ \vec{b}_{2n} \end{pmatrix} = \mathbf{M}^{-1} \cdot \begin{pmatrix} \vec{f}_{n+1} - \mu \vec{f}_1 \\ \vdots \\ \vec{f}_{2n} - \mu \vec{f}_n \end{pmatrix}, \begin{pmatrix} \vec{b}_{n+1}^* \\ \vdots \\ \vec{b}_{2n}^* \end{pmatrix} = \mathbf{M}^{\top} \cdot \begin{pmatrix} \vec{f}_{n+1}^* \\ \vdots \\ \vec{f}_{2n}^* \end{pmatrix}.
$$

Then it implicitly defines $\mathbb{B} = \{\vec{b}_1, ..., \vec{b}_{2n+2}\}$, $\mathbb{B}^* = \{\vec{b}_1^*, ..., \vec{b}_{2n+2}^*\}$, $\mathbb{D} = \{\vec{d}_1, ..., \vec{d}_5\}$, $\mathbb{D}^* = \{\vec{d}_1^*, ..., \vec{d}_5^*\}$, and sets $\widehat{\mathbb{B}} = \{\vec{b}_1, ..., \vec{b}_{2n}, \vec{b}_{2n+2}\}$, $\widehat{\mathbb{B}}^* = \{\vec{b}_1^*, ..., \vec{b}_n^*, \vec{b}_{2n+1}^*\}$, $\widehat{\mathbb{D}} = \{\vec{d}_1, \vec{d}_2, \vec{d}_4\}$, $\widehat{\mathbb{D}}^* = \{\vec{d}_1^*, \vec{d}_3^*\}$.

Finally, $\mathcal{B}$ gives the public parameters $\widetilde{\mathrm{PP}} = (q, \mathbb{V}_1, \mathbb{V}_1^*, \mathbb{V}_2, \mathbb{V}_2^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, \mathbb{A}_2, \mathbb{A}_2^*), E_1, E_2)$ to $\mathcal{A}$. The master secret key, $\widetilde{\mathrm{MSK}} = (\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$, is known to $\mathcal{B}$.

Key queries: To answer key queries made by $\mathcal{A}$, $\mathcal{B}$ randomly chooses $\beta, \eta, \eta' \xleftarrow{\$} \mathbb{Z}_q$ and responds as follows:

$$
\begin{aligned}
k_1^* &= g_2^{\beta \sum_{i=1}^{n} y_i \vec{f}_i^* + \eta \vec{f}_{2n+1}^*} (g_2^{\mu})^{\beta \sum_{i=1}^{n} y_i \vec{f}_{n+i}^*} \\
&= g_2^{\beta \sum_{i=1}^{n} y_i (\vec{f}_i^* + \mu \vec{f}_{n+i}^*) + \eta \vec{f}_{2n+1}^*} \\
&= g_2^{\beta \sum_{i=1}^{n} y_i \vec{b}_i^* + \eta \vec{b}_{2n+1}^*}, \\
k_2^* &= g_2^{\beta \vec{h}_1^* + \eta' \vec{h}_3^*} (g_2^{\mu})^{\beta \vec{h}_2^*} \\
&= g_2^{\beta(\vec{h}_1^* + \mu \vec{h}_2^*) + \eta' \vec{h}_3^*} \\
&= g_2^{\beta \vec{d}_1^* + \eta' \vec{d}_3^*}.
\end{aligned}
$$

Simulated ciphertext: To answer ciphertext query that $\mathcal{A}$ makes, $\mathcal{B}$ chooses $\widehat{\alpha}, \alpha^{'}, \varphi, \varphi^{'} \xleftarrow{\$} \mathbb{Z}_q$, implicitly sets $\alpha = \widehat{\alpha} + \alpha^{'}\mu$ and responds with

$$\begin{aligned} c_1 &= g_1^{\widehat{\alpha}\sum_{i=1}^{n} x_i \vec{f}_i + \alpha^{'}\sum_{i=1}^{n} x_i \vec{f}_{n+i} + \varphi \vec{f}_{2n+2}} \\ &= g_1^{(\widehat{\alpha}+\alpha^{'}\mu)\sum_{i=1}^{n} x_i \vec{f}_i + \alpha^{'}\sum_{i=1}^{n} x_i (\vec{f}_{n+i} - \mu \vec{f}_i) + \varphi \vec{f}_{2n+2}} \\ &= g_1^{\alpha\sum_{i=1}^{n} x_i \vec{b}_i + \alpha^{'}\sum_{i=1}^{n} t_i \vec{b}_{n+i} + \varphi \vec{b}_{2n+2}}, \\ c_2 &= g_1^{\widehat{\alpha}\vec{h}_1 + \alpha^{'}\vec{h}_2 + \varphi^{'}\vec{h}_4} \\ &= g_1^{(\widehat{\alpha}+\alpha^{'}\mu)\vec{h}_1 + \alpha^{'}(\vec{h}_2 - \mu\vec{h}_1) + \varphi^{'}\vec{h}_4} \\ &= g_1^{\alpha\vec{d}_1 + \alpha^{'}\vec{d}_2 + \varphi^{'}\vec{h}_4}, \end{aligned}$$

where $\vec{t} = \vec{x}\mathbf{M}$.

The output of $\mathcal{B}$'s setup has the same distribution of the corresponding output of $\widetilde{\textbf{Setup}}$ and thus like in $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$. Similarly for the answer to key queries. $\vec{t}$ is distributed uniformly and independently in $\mathbb{Z}_q^n \setminus \{\vec{0}\}$ because of $\mathbf{M} \xleftarrow{\$} GL(n, \mathbb{Z}_q)$ and $\vec{x} \neq \vec{0}$. The output of $\mathcal{B}$'s simulated ciphertext is distributed like the output of algorithm **Encrypt** and thus exactly as in $H_1^{\mathcal{A}}$ and $H_2^{\mathcal{A}}$. Thus the view of $\mathcal{A}$ in $H_1^{\mathcal{A}}$ and that in $H_2^{\mathcal{A}}$ have the same distribution. □

We prove that, for all PPT adversaries $\mathcal{A}$, $H_2^{\mathcal{A}}$ is computationally indistinguishable from $Ideal^{\mathcal{A}}$ under the SXDH assumption in lemma 4.

Lemma 4. Assuming the existence of the SXDH assumption, for all PPT adversaries $\mathcal{A}$, $H_2^{\mathcal{A}} \approx_c Ideal^{\mathcal{A}}$.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish between $H_2^{\mathcal{A}}$ and $Ideal^{\mathcal{A}}$. Then, we construct a PPT algorithm $\mathcal{B}$ which is given an instance of the SXDH assumption $\mathcal{G}_{\sigma}^{\mathrm{SXDH}}(1^{\lambda}) = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), g_2^{\mu}, g_2^{\nu}, Y_{\sigma})$ for $\sigma \in \{0, 1\}$. It is $\mathcal{B}$'s task to decide whether Y_{σ} is distributed as $Y_0 = g_2^{\mu\nu}$ or as $Y_1 = g_2^{\mu\nu+r}$, where $r \xleftarrow{\$} \mathbb{Z}_q \setminus 0$.

Setup: $\mathcal{B}$ samples dual orthonormal bases $(\mathbb{F} = \{\vec{f}_1, ..., \vec{f}_{2n+2}\}, \mathbb{F}^* = \{\vec{f}_1, ..., \vec{f}_{2n+2}^*\}) \xleftarrow{\$} \mathcal{G}_{ob}(1^{\lambda}, \mathbb{Z}_q^{2n+2})$ and $(\mathbb{H} = \{\vec{h}_1, ..., \vec{h}_4\}, \mathbb{H}^* = \{\vec{h}_1^*, ..., \vec{h}_4^*\}) \xleftarrow{\$} \mathcal{G}_{ob}(1^{\lambda}, \mathbb{Z}_q^4)$. It chooses $\mathbf{M} \xleftarrow{\$} GL(n, \mathbb{Z}_q)$ and sets

$$\begin{aligned} &\vec{b}_i = \vec{f}_i (i = 1, ..., n), && \vec{b}_i^* = \vec{f}_i^* + \mu \vec{f}_{n+i}^* (i = 1, ..., n), \\ &\vec{b}_{2n+i} = \vec{f}_{2n+i} (i = 1, 2), && \vec{b}_{2n+i}^* = \vec{f}_{2n+i}^* (i = 1, 2), \\ &\vec{d}_i = \vec{h}_i (i = 1, 3, 4), && \vec{d}_i^* = \vec{h}_i^* (i = 2, 3, 4), \\ &\vec{d}_2 = \vec{h}_2 - \mu\vec{h}_1, && \vec{d}_1^* = \vec{h}_1^* + \mu\vec{h}_2^*, \end{aligned}$$

$$\begin{pmatrix} \vec{b}_{n+1} \\ \vdots \\ \vec{b}_{2n} \end{pmatrix} = \mathbf{M}^{-1} \cdot \begin{pmatrix} \vec{f}_{n+1} - \mu\vec{f}_1 \\ \vdots \\ \vec{f}_{2n} - \mu\vec{f}_n \end{pmatrix}, \begin{pmatrix} \vec{b}_{n+1}^* \\ \vdots \\ \vec{b}_{2n}^* \end{pmatrix} = \mathbf{M}^{\top} \cdot \begin{pmatrix} \vec{f}_{n+1}^* \\ \vdots \\ \vec{f}_{2n}^* \end{pmatrix}.$$

Then it implicitly defines $\mathbb{B} = \{\vec{b}_1, ..., \vec{b}_{2n+2}\}$, $\mathbb{B}^* = \{\vec{b}_1^*, ..., \vec{b}_{2n+2}^*\}$, $\mathbb{D} = \{\vec{d}_1, ..., \vec{d}_5\}$, $\mathbb{D}^* = \{\vec{d}_1^*, ..., \vec{d}_5^*\}$, and sets $\widehat{\mathbb{B}} = \{\vec{b}_1, ..., \vec{b}_{2n}, \vec{b}_{2n+2}\}$, $\widehat{\mathbb{B}}^* = \{\vec{b}_1^*, ..., \vec{b}_n^*, \vec{b}_{2n+1}^*\}$, $\widehat{\mathbb{D}} = \{\vec{d}_1, \vec{d}_2, \vec{d}_4\}$,$\widehat{\mathbb{D}}^* = \{\vec{d}_1^*, \vec{d}_3^*\}$.

Finally, $\mathcal{B}$ give $\mathcal{A}$ the public parameters $\widetilde{\mathrm{PP}} = (q, \mathbb{V}_1, \mathbb{V}_1^*, \mathbb{V}_2, \mathbb{V}_2^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, \mathbb{A}_2, \mathbb{A}_2^*), E_1, E_2)$. The master secret key, $\widetilde{\mathrm{MSK}} = (\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*)$, is known to $\mathcal{B}$.

First stage key queries: To answer key queries made by $\mathcal{A}$, $\mathcal{B}$ chooses random $\beta, \eta, \eta^{'} \xleftarrow{\$} \mathbb{Z}_q$ and responds as follows:

$$\begin{aligned} k_1^* &= g_2^{\beta\sum_{i=1}^{n} y_i \vec{f}_i^* + \eta\vec{f}_{2n+1}^*} (g_2^{\mu})^{\beta\sum_{i=1}^{n} y_i \vec{f}_{n+i}^*} \\ &= g_2^{\beta\sum_{i=1}^{n} y_i (\vec{f}_i^* + \mu\vec{f}_{n+i}^*) + \eta\vec{f}_{2n+1}^*} \\ &= g_2^{\beta\sum_{i=1}^{n} y_i \vec{b}_i^* + \eta\vec{b}_{2n+1}^*}, \\ k_2^* &= g_2^{\beta\vec{h}_1^* + \eta^{'}\vec{h}_3^*} (g_2^{\mu})^{\beta\vec{h}_2^*} \\ &= g_2^{\beta(\vec{h}_1^* + \mu\vec{h}_2^*) + \eta^{'}\vec{h}_3^*} \\ &= g_2^{\beta\vec{d}_1^* + \eta^{'}\vec{d}_3^*}. \end{aligned}$$

Simulated ciphertext: To answer ciphertext query that $\mathcal{A}$ makes, $\mathcal{B}$ chooses $\widehat{\alpha}, \alpha^{'}, \varphi, \varphi^{'} \xleftarrow{\$} \mathbb{Z}_q$, implicitly sets $\alpha = \widehat{\alpha} + \alpha^{'}\mu$, and responds with

$$c_1 = g_1^{\widehat{\alpha}\sum_{i=1}^{n} x_i \vec{f}_i + \alpha^{'}\sum_{i=1}^{n} x_i \vec{f}_{n+i} + \varphi \vec{f}_{2n+2}}$$

$$
\begin{aligned}
&= g_1^{(\widehat{\alpha}+\alpha^{'}\mu)\sum_{i=1}^{n} x_i\vec{f}_i+\alpha^{'}\sum_{i=1}^{n} x_i(\vec{f}_{n+i}-\mu\vec{f}_i)+\varphi\vec{f}_{2n+2}}\\
&= g_1^{\alpha\sum_{i=1}^{n} x_i\vec{b}_i+\alpha^{'}\sum_{i=1}^{n} t_i\vec{b}_{n+i}+\varphi\vec{b}_{2n+2}},\\
c_2 &= g_1^{\widehat{\alpha}\vec{h}_1+\alpha^{'}\vec{h}_2+\varphi^{'}\vec{h}_4}\\
&= g_1^{(\widehat{\alpha}+\alpha^{'}\mu)\vec{h}_1+\alpha^{'}(\vec{h}_2-\mu\vec{h}_1)+\varphi^{'}\vec{h}_4}\\
&= g_1^{\alpha\vec{d}_1+\alpha^{'}\vec{d}_2+\varphi^{'}\vec{h}_4}.
\end{aligned}
$$

where $\vec{t}=\vec{x}\mathbf{M}$.

Second stage key queries: In what follows, we will assume by implication that the st', which is transfered to its next query, is updated by the simulator. First of all, $\mathcal{B}$ chooses a random $k=\{1,...,p_2\}$, where p_2 is the number of the key queries asked by the adversary $\mathcal{A}$ in the second stage. $\mathcal{B}$ responds to $\mathcal{A}$'s i-th key query of the second stage for vector $\vec{y}$ by distinguishing between the following three cases.

- When $i<k$, $\mathcal{B}$ computes and answers

$$
\begin{aligned}
k_1^* &= g_2^{\beta\sum_{i=1}^{n} y_i\vec{f}_i^*+\beta'\sum_{i=1}^{n} y_i\vec{f}_{n+i}^*+\eta\vec{f}_{2n+1}^*}(g_2^{\mu})^{\beta\sum_{i=1}^{n} y_i\vec{f}_{n+i}^*}\\
&= g_2^{\beta\sum_{i=1}^{n} y_i(\vec{f}_i^*+\mu\vec{f}_{n+i}^*)+\beta'\sum_{i=1}^{n} y_i\vec{f}_{n+i}^*+\eta\vec{f}_{2n+1}^*}\\
&= g_2^{\beta\sum_{i=1}^{n} y_i\vec{b}_i^*+\beta^{'}\sum_{i=1}^{n} w_i\vec{b}_{n+i}^*+\eta\vec{b}_{2n+1}^*},\\
k_2^* &= g_2^{\beta\vec{h}_1^*+\beta^{'}\vec{h}_2^*+\eta^{'}\vec{h}_3^*}(g_2^{\mu})^{\beta\vec{h}_2^*}\\
&= g_2^{\beta(\vec{h}_1^*+\mu\vec{h}_2^*)+\beta^{'}\vec{h}_2^*+\eta^{'}\vec{h}_3^*}\\
&= g_2^{\beta\vec{d}_1^*+\beta^{'}\vec{d}_2^*+\eta^{'}\vec{d}_3^*},
\end{aligned}
$$

where $\beta,\beta^{'},\eta,\eta^{'}\xleftarrow{\$}\mathbb{Z}_q$ and $\vec{w}=\vec{y}(\mathbf{M}^{\top})^{-1}$.

- When $i=k$, $\mathcal{B}$ computes and answers as following:

$$
\begin{aligned}
k_1^* &= (g_2^{\nu})^{\sum_{i=1}^{n} y_i\vec{f}_i^*}(Y_\sigma)^{\sum_{i=1}^{n} y_i\vec{f}_{n+i}^*}g_2^{\eta\vec{f}_{2n+1}^*}\\
&= g_2^{\nu\sum_{i=1}^{n} y_i(\vec{f}_i^*+\mu\vec{f}_{n+i}^*)+r\sum_{i=1}^{n} y_i\vec{f}_{n+i}^*+\eta\vec{f}_{2n+1}^*}\\
&= g_2^{\nu\sum_{i=1}^{n} y_i\vec{b}_i^*+r\sum_{i=1}^{n} w_i\vec{b}_{n+i}^*+\eta\vec{b}_{2n+1}^*},\\
k_2^* &= (g_2^{\nu})^{\vec{h}_1^*}(Y_\sigma)^{\vec{h}_2^*}g_2^{\eta^{'}\vec{h}_3^*}\\
&= g_2^{\nu(\vec{h}_1^*+\mu\vec{h}_2^*)+r\vec{h}_2^*+\eta^{'}\vec{h}_3^*}\\
&= g_2^{\nu\vec{d}_1^*+r\vec{d}_2^*+\eta^{'}\vec{d}_3^*},
\end{aligned}
$$

where $\beta,\beta^{'},\eta,\eta^{'}\xleftarrow{\$}\mathbb{Z}_q$ and $\vec{w}=\vec{y}(\mathbf{M}^{\top})^{-1}$.

- When $i>k$, $\mathcal{B}$ answers a normal form of the master secret key as following:

$$
\begin{aligned}
k_1^* &= g_2^{\beta\sum_{i=1}^{n} y_i\vec{f}_i^*+\eta\vec{f}_{2n+1}^*}(g_2^{\mu})^{\beta\sum_{i=1}^{n} y_i\vec{f}_{n+i}^*}\\
&= g_2^{\beta\sum_{i=1}^{n} y_i(\vec{f}_i^*+\mu\vec{f}_{n+i}^*)+\eta\vec{f}_{2n+1}^*}\\
&= g_2^{\beta\sum_{i=1}^{n} y_i\vec{b}_i^*+\eta\vec{b}_{2n+1}^*},\\
k_2^* &= g_2^{\beta\vec{h}_1^*+\eta^{'}\vec{h}_3^*}(g_2^{\mu})^{\beta\vec{h}_2^*}\\
&= g_2^{\beta(\vec{h}_1^*+\mu\vec{h}_2^*)+\eta^{'}\vec{h}_3^*}\\
&= g_2^{\beta\vec{d}_1^*+\eta^{'}\vec{d}_3^*},
\end{aligned}
$$

where $\beta,\eta,\eta^{'}\xleftarrow{\$}\mathbb{Z}_q$.

We consider $\mathcal{A}$'s view while interacting with $\mathcal{B}$. The distribution of the public parameters is the same as that of the corresponding output of $\widetilde{\textbf{Setup}}$. The answer of the first stage queries has the same distribution of the output of algorithm **KeyGen**. The distribution of the ciphertext constructed by $\mathcal{B}$ equals that of the output of $\widetilde{\textbf{Encrypt}}$. Then, we consider the answers of the second stage queries. The matching queries and the first $k-1$ unmatched queries are distributed as the

output of $\widetilde{\mathbf{KeyGen}}(.,\langle\vec{x},.\rangle,k)$ and $\widetilde{\mathbf{KeyGen}}(.,\langle\vec{x},.\rangle,k-1)$. In the same way, the distribute of the last p_2-k unmatched queries is the same as the output of $\widetilde{\mathbf{KeyGen}}(.,\langle\vec{x},.\rangle,k)$ and $\widetilde{\mathbf{KeyGen}}(.,\langle\vec{x},.\rangle,k-1)$. If $\sigma=0$, i.e., $Y_0=g_2^{\mu\nu}$, then the answer to the k-th query is distributed in term of the output of $\widetilde{\mathbf{KeyGen}}(\mathrm{MSK},\langle\vec{x},.\rangle,k)$, and If $\sigma=1$, i.e., $Y_1=g_2^{\mu\nu+r}$, where $r\xleftarrow{\$}\mathbb{Z}_q\setminus 0$, then the answer to the k-th query is distributed in term of the output of $\widetilde{\mathbf{KeyGen}}(\mathrm{MSK},\langle\vec{x},.\rangle,k-1)$.

Next, we consider the joint distribution of c_1 and k_1^*. $\mathcal{B}$ runs algorithm $\widetilde{\mathbf{Encrypt}}$ and computes $c_1=g_1^{\alpha\sum_{i=1}^n x_i\vec{b}_i+\alpha^{'}\sum_{i=1}^n\langle\vec{x},\vec{m}_i\rangle\vec{b}_{n+i}+\varphi\vec{b}_{2n+2}}$, where $\vec{m}_i=(\mathbf{M}_{1,i},...,\mathbf{M}_{n,i})$. Coefficients, $\alpha,\varphi\in\mathbb{Z}_q$ and $(\alpha^{'}\langle\vec{x},\vec{m}_1\rangle,...,\alpha^{'}\langle\vec{x},\vec{m}_n\rangle)\in\mathbb{Z}_q^n\setminus\{\vec{0}\}$, are distributed uniformly and independently.

When $r=0$, $\mathcal{B}$ runs algorithm $\mathbf{KeyGen}$ and computes $k_1^*=g_2^{\nu\sum_{i=1}^n y_i\vec{b}_i^*+\eta\vec{b}_{2n+1}^*}$. Then, ν,η and the above coefficients in c_1 are independently uniform. Therefore, the joint distribution of c_1 and k_1^* has the equivalence of that of the normal ciphertext and the normal secret key.

When $r\xleftarrow{\$}\mathbb{Z}_q\setminus 0$, $\mathcal{B}$ runs algorithm $\widetilde{\mathbf{KeyGen}}$ and computes $k_1^*=g_2^{\nu\sum_{i=1}^n y_i\vec{b}_i^*+r\sum_{i=1}^n\langle\vec{y},\vec{m}_i^{'}\rangle\vec{b}_{n+i}^*+\eta\vec{b}_{2n+1}^*}$, where $\vec{m}_i^{'}=((\mathbf{M}^\top)_{1,i}^{-1},...,(\mathbf{M}^\top)_{n,i}^{-1})$. Then, ν,η and the other coefficients in c_1 and k_1^* are independently uniform. Coefficients $(\alpha^{'}\langle\vec{x},\vec{m}_1\rangle,...,\alpha^{'}\langle\vec{x},\vec{m}_n\rangle)\in\mathbb{Z}_q^n\setminus\{\vec{0}\}$ in c_1 and coefficients $(r\langle\vec{y},\vec{m}_1^{'}\rangle,...,r\langle\vec{y},\vec{m}_n^{'}\rangle)\in\mathbb{Z}_q^n\setminus\{\vec{0}\}$ in k_1^* are independently and uniformly distributed from Lemma 1. Therefore, the joint distributed of c_1 and k_1^* has the equivalence of that of the simulated ciphertext and the normal secret key. □

Remark 3. We also can prove the SIPE sheme is many-AD-SIM-secure if the adversary gets hold of an unbounded number of ciphertexts and secret keys. The solution is to sample different dual orthonormal bases for each message.

Remark 4. Our construction can achieve security without any leakage beyond the ideal functionality. We consider the case where the adversary outputs two challenge plaintexts $\vec{x}_1,\vec{x}_2$ and makes a single key query for vector $\vec{y}$. For indistinguishability-based security to be satisfiable the ideal functionality imposes the only constraint that is $\langle\vec{x}_1,\vec{y}\rangle=\langle\vec{x}_2,\vec{y}\rangle$. In line with the simulation-based security notion, we would only require $\langle\vec{x}_1,\vec{y}\rangle\approx_c\langle\vec{x}_2,\vec{y}\rangle$ instead of $\langle\vec{x}_1,\vec{y}\rangle=\langle\vec{x}_2,\vec{y}\rangle$. It is straightforward to see that one-AD-SIM security does provide such a guarantee. Moreover, there is not attack where the adversary tries to decrypt normal ciphertexts for several times, because the adversary will not be able to produce normal ciphertexts by itself in the private-key setting.

References

1 De Caro A, Iovino V, Jain A, et al. On the achievability of simulation-based security for functional encryption. In:Proceedings of the 33rd Annual Cryptology Conference, Santa Barbara, 2013. 519-535

2 Boneh D, Sahai A, Waters B. Functional encryption: definitions and challenges. In: Proceedings of the 8th Conference on Theory of Cryptography, Providence, 2011. 253-273

3 O'Neill A. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. http://eprint.iacr.org/2010/556

4 Agrawal S, Gorbunov S, Vaikuntanathan V, et al. Functional encryption: new perspectives and lower bounds. In:Proceedings of the 33rd Annual Cryptology Conference, Santa Barbara, 2013. 500-518

5 Kim S, Lewi K, Mandal A, et al. Function-hiding inner product encryption is practical. Cryptology ePrint Archive, Report 2016/440, 2016. http://eprint.iacr.org/2016/440

6 Agrawal S, Koppula V, Waters B. Impossibility of simulation secure functional encryption even with random oracles. Cryptology ePrint Archive, Report /2016/959, 2016. http://eprint.iacr.org/2016/959

7 Iovino V, Żebroski K. Simulation-based secure functional encryption in the random oracle model. In:Proceedings of the 4th International Conference on Progress in Cryptology, Bienvenido, 2015. 21-39

8 Bishop A, Jain A, Kowalczyk L. Function-hiding inner product encryption. In:Proceedings of the 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, 2015. 470-491

9 Datta P, Dutta R, Mukhopadhyay S. Functional encryption for inner product with full function privacy. In:Proceedings of the 19th International Conference on the Theory and Practice of Public-Key Cryptography, Taipei, 2016. 164-195

10 Lindell Y. How to simulate it - a tutorial on the simulation proof technique. Cryptology ePrint Archive, Report 2016/046, 2016. http://eprint.iacr.org/2016/046

11 Waters B. Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In:Proceedings of the 29th Annual International Cryptology Conference, Santa Barbara, 2009. 619-636

12 Okamoto T, Takashima K. Fully secure functional encryption with general relations from the decisional linear assumption. In:Proceedings of the 30th Annual Cryptology Conference. Santa Barbara, 2010. 191-208