Hindawi
Security and Communication Networks
Volume 2018, Article ID 6561418, 12 pages
https://doi.org/10.1155/2018/6561418

Research Article

Improved Construction for Inner Product Functional Encryption

Qingsong Zhao,[1,2,3] Qingkai Zeng,[1,2] and Ximeng Liu[4,5]

[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]*College of Information Science and Technology, Nanjing Agricultural University, Nanjing 210095, China*
[4]*College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350117, China*
[5]*School of Information Systems, Singapore Management University, 178902, Singapore*

Correspondence should be addressed to Qingkai Zeng; zqk@nju.edu.cn

Received 4 February 2018; Revised 30 June 2018; Accepted 22 July 2018; Published 13 August 2018

Academic Editor: Petros Nicopolitidis

Functional encryption (FE) is a vast new paradigm for encryption scheme which allows tremendous flexibility in accessing encrypted data. In a FE scheme, a user can learn specific function of encrypted messages by restricted functional key and reveals nothing else about the messages. Besides the standard notion of data privacy in FE, it should protect the privacy of the function itself which is also crucial for practical applications. In this paper, we construct a secret key FE scheme for the inner product functionality using asymmetric bilinear pairing groups of prime order. Compared with the existing similar schemes, our construction reduces both necessary storage and computational complexity by a factor of 2 or more. It achieves simulation-based security, security strength which is higher than that of indistinguishability-based security, against adversaries who get hold of an unbounded number of ciphertext queries and adaptive secret key queries under the External Decisional Linear (XDLIN) assumption in the standard model. In addition, we implement the secret key inner product scheme and compare the performance with the similar schemes.

1. Introduction

Traditional public-key encryption provides all-or-nothing access to data: you can either recover the entire plaintext or reveal nothing from the ciphertext. Functional encryption (FE) [1–3] is a vast new paradigm for encryption scheme which allows tremendous flexibility in accessing encrypted data. In a FE scheme, a secret key sk_f embedded with a function f can be created from a master secret key msk. Then, given a ciphertext for x, a user learns $f(x)$ and reveals nothing else about x. In recent years, the cryptographic community has made great progress in research on the security of FE and constructions for such schemes (see, for instance, [4–11] and any more).

There are two notions of security for a FE scheme, i.e., indistinguishability-based security and simulation-based security. The former one requires that an adversary cannot distinguish between ciphertexts of any two messages x_0, x_1 with access to a secret key sk_f for a function f such that $f(x_0) = f(x_1)$. In contrast, the latter one requires that the view of the adversary can be simulated by a simulator, given only access to the secret keys and the function evaluated on the corresponding messages. Note that simulation-based security has higher security strength than indistinguishability-based security such that there exists an indistinguishability-based secure FE scheme for a certain functionality which is not able to be proved secure under simulation-based security [1, 3].

The traditional FE only considers data privacy and omits to protect the privacy of the function itself which is also crucial for practical applications. Consider the case where Bob wants to store his files in a cloud. Before uploading his files to the cloud, he employs a FE scheme to encrypt them avoiding leakage of data privacy and then he uploads the encryption form to the cloud. Later on, Bob wants to query his data by offering the cloud a key sk_f for a function f of his choice. However, if the FE scheme cannot support the privacy for the function, the key sk_f may reveal Bob's query f

Table 1: Comparison of our SSSK-IPE scheme. n is the dimension of vector. We have eight security notions xx-yy-zzz, where xx ∈ {one, many} refers to a single or multiple challenge ciphertexts; yy ∈ {SEL, AD} refers to selectively or adaptively chosen ciphertext queries; zzz ∈ {IND, SIM} refers to indistinguishability-based vs simulation-based security.

Scheme	Master secret key	Ciphertext Secret key	Scalar multiplications	Pairing operations	Assum.	Security
BJK15[16]	$8n^2 + 8$	$2n + 2$	$2n + 2$	$2n + 2$	SXDH	many-AD-IND
DDM16[17]	$8n^2 + 12n + 28$	$4n + 8$	$4n + 8$	$4n + 8$	SXDH	many-AD-IND
TAO16 [18]	$4n^2 + 18n + 20$	$2n + 5$	$2n + 5$	$2n + 5$	XDLIN	many-AD-IND
ZZL17 [19]	$6n^2 + 10n + 24$	$2n + 4$	$2n + 4$	$2n + 4$	SXDH	many-AD-SIM
This work	$2n^2 + 18n + 36$	$n + 6$	$n + 6$	$n + 6$	XDLIN	many-AD-SIM

entirely to the cloud, which is not desirable when the function includes confidential information.

Due to the importance, some works focus on function privacy of FE, and this was first studied in [12] in the secret key setting. This is later followed by the work of [5, 13] in the secret key setting and that of [14, 15] in the public-key setting. During the two scenarios of the public-key setting and the secret key setting, the degree to which function privacy can be satisfied differs dramatically. Specifically, a public-key FE scheme is inherent in leaking confidential information about the function. Note that an attacker who holds a secret key sk_f can always generate, on its own, the ciphertext for x_i for message x_i of her choice and then use sk_f to learn $f(x_i)$. This can reveal nontrivial information about the function f. On the other hand, since an attacker holding a secret key sk_f cannot encrypt new messages in the secret key setting, such kind of attack is no longer applied.

1.1. Functional Encryption for Inner Product. Although FE supports the computation of general circuits relying on a wide spectrum of assumptions, there are two major problems with the state-of-the-art general FE constructions. First, the security of some constructions is only ensured so long as the adversary gets hold of a priori bounded number of secret keys [9, 10, 20]. Second, some solutions rely on tools such as multilinear maps [21] and indistinguishability obfuscation [8, 22] which are both impractical and founded on new security assumption undergone minimal scrutiny. This inspires us to explore constructions for firsthand and effective FE schemes for functionalities which focus on the inner product functionality as a first attempt [16–19, 23–26].

In an inner product encryption (IPE) scheme, a ciphertext $ct_{\vec{x}}$ is related to a vector $\vec{x} \in \mathbb{Z}_q^n$ of length n and a secret key $sk_{\vec{y}}$ to a vector $\vec{y} \in \mathbb{Z}_q^n$ of length n. Given the ciphertext and the secret key, the decryption algorithm computes the inner product $\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{n} x_i y_i$. Notice that the formulation of IPE is distinct from that of inner product predicate encryption in [12, 13, 27–30]. The ciphertext for a message m in an inner product predicate encryption scheme comes along with an attribute $\vec{x} \in \mathbb{Z}_q^n$, and a secret key corresponds to a vector $\vec{y} \in \mathbb{Z}_q^n$. When the ciphertext with $\vec{x}$ is decrypted with the secret key for $\vec{y}$, the decryption algorithm outputs m iff $\langle \vec{x}, \vec{y} \rangle = 0$. By contrast, the output in the IPE formulation is the actual value of the inner product. In this paper, we consider functional privacy in inner product encryption, i.e., secret key inner product encryption.

1.2. Related Work. Abdalla et al. [23] presented a direct construction of public-key IPE under an indistinguishability-based definition. The construction is only proved to be secure against selective adversaries which are asked to commit to their challenges at the beginning of the security game. Following work [24] presented adaptively secure schemes where the messages x_0 and x_1 may be adaptively chosen at any point in time, based on the previously collected information. Bishop et al. [16] proposed a function-hiding IPE scheme under the Symmetric External Diffie-Hellman (SXDH) assumption, which satisfies an indistinguishability-based definition, and considered adaptive adversaries. However, the scheme is available in a rather weak security model which places limit on adversaries' queries. Specially, all ciphertext queries $\vec{x}_0, \vec{x}_1$ and all secret key queries $\vec{y}_0, \vec{y}_1$ are constrained by $\langle \vec{x}_0, \vec{y}_0 \rangle = \langle \vec{x}_0, \vec{y}_1 \rangle = \langle \vec{x}_1, \vec{y}_0 \rangle = \langle \vec{x}_1, \vec{y}_1 \rangle$. The constraint obviously weakens the security of the scheme, and this is in violation of the intuitive spirit of function privacy. Recently, Datta et al. [17] developed a function-hiding IPE scheme under the SXDH assumption where the restriction on adversaries' queries is only $\langle \vec{x}_0, \vec{y}_0 \rangle = \langle \vec{x}_1, \vec{y}_1 \rangle$. Tomida et al. [18] constructed a more efficient function-hiding IPE scheme than that of [17] under the External Decisional Linear (XDLIN) assumption. Kim et al. [25] put forth a fully-secure function-hiding IPE scheme with less parameter sizes and run time complexity than in [16, 17]. The scheme is proved simulation-based secure in the generic model of bilinear maps. For the first time Zhao et al. [19] presented a simulation-based secure secret key IPE scheme under the SXDH assumption in the standard model. The scheme can tolerate an unbounded number of ciphertext queries and adaptive key queries.

1.3. Our Contribution. We construct an efficient simulation-based secure secret key IPE (SSSK-IPE) scheme in the standard model. We compare our scheme with related works in Table 1 where group exponentiations on cyclic groups are involved in key generation algorithm and encryption algorithm, and pairing operations on bilinear pairing groups are involved in decryption algorithm. We achieve an outstanding reduction by a factor of 2 or more in computational complexity. Our scheme achieves $n + 6$ group elements

in secret key and ciphtertext, which also reduces storage complexity by a factor of 2 or more. Hence, performance in the SSSK-IPE scheme is superior to that in the previous schemes in both storage complexity and computation complexity. Furthermore, our scheme is based on the XDLIN assumption which is weaker than the SXDH assumption. In more detail, the SXDH assumption relies on type 3 bilinear pairing groups, while the XDLIN assumption relies on any type of bilinear pairing groups [18]. Therefore from this angle, the SXDH assumption is stronger than the XDLIN assumption. Although the construction of [18] was proved to be indistinguishability-based secure under the XDLIN assumption and also succeeded in improving efficiency, both storage complexity and computation complexity of our scheme are better than that of [18] and our scheme achieves simulation-base security, security strength of which is higher than that of indistinguishability-based security. In addition, we implement our SSSK-IPE scheme and compare the performance with the similar schemes in Section 5.

To guarantee correctness, our scheme requires that inner products be within a range of polynomial-size, which is consistent with other schemes in Table 1. As pointed out in [16], this is reasonable for statistical computations because the computations, like the average over a polynomial-size database, will naturally be contained within a polynomial range. Furthermore, our scheme is simulation-based secure against adversaries who hold in an unbounded number of ciphertext queries and adaptive key queries. Although very basic functionalities such as IBE are simulation-based secure for a priori bounded number of ciphertext queries in the standard model [1, 3, 4], this is possible for an unbounded number of ciphertext queries if adversaries have an underlying polynomial-size range.

1.4. Technical Overview. Our SSSK-IPE scheme uses dual pairing vector spaces (DPVS) to construct, as in [16–19], which is brought forward by Okamoto et al. [31, 32]. DPVS has the features of hidden linear subspaces in prime order bilinear group setting. A DPVS of dimension $n + 6$ is introduced in our construction, where n is the dimension of inner product vectors. Typically, we sample a pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ and only use the $n + 4$ dimension $\mathbb{B}$ and the $n + 4$ dimension $\mathbb{B}^*$ to encode vector $\vec{x}$ and vector $\vec{y}$, respectively. Compared with the previous schemes, our scheme at least saves n dimensions of vector spaces. We preserve the remaining hidden dimensions of $\mathbb{B}$ and $\mathbb{B}^*$ for the security reduction. Specially, between two hybrid experiments, a hidden dimension can be used for reducing a difference of one coefficient in a secret key or a cipertext to a XDLIN instance.

2. Preliminaries

Let λ be the security parameter. If S is a set, $x \xleftarrow{\$} S$ denotes the process of choosing uniformly at random from S. Let $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ be distribution ensembles. We say that $X \approx_c Y$ are computationally indistinguishable between X and Y, if for all nonuniform probabilistic polynomial time D and every $n \in \mathbb{N}$, the difference between $\Pr[D(X_n) = 1]$ and $\Pr[D(Y_n) = 1]$ is negligible. Let $negl(\lambda)$ be a negligible function in λ. Moreover, we write $\vec{x}$ to denote a vector $(x_1, \ldots, x_n) \in \mathbb{Z}_q^n$ of length n for some positive integer q and n. We use $\langle \vec{a}, \vec{b} \rangle$ to denote the inner product, $\sum_{i=1}^{n} a_i b_i \bmod q$, of vectors $\vec{a} \in \mathbb{Z}_q^n$ and $\vec{b} \in \mathbb{Z}_q^n$. We use upper case boldface to denote matrices. $\mathbf{X}^T$ denotes transpose of the matrix $\mathbf{X}$. $GL(n, \mathbb{Z}_q)$ denotes the general linear group of degree n over $\mathbb{Z}_q$. $\mathbb{Z}_q^\times$ denotes a set of integers $\{1, \ldots, q-1\}$.

2.1. Definition of SSSK-IPE. We introduce the definition of simulation-based secure secret key IPE (SSSK-IPE).

Definition 1 (SSSK-IPE). A SSSK-IPE scheme is composed of the four PPT algorithms defined below:

(i) **SSSK-IPE.Setup** $(1^\lambda, n) \longrightarrow (msk, pp)$: The setup algorithm receives as input the security parameters λ and n, being vector length, and outputs a master secret key msk and public parameters pp.

(ii) **SSSK-IPE.Encrypt** $(msk, pp, \vec{x}) \longrightarrow ct_{\vec{x}}$: The encryption algorithm receives as input the master secret key msk, the public parameters pp, and a vector $\vec{x} \in \mathbb{Z}_q^n$ and outputs a ciphertext $ct_{\vec{x}}$.

(iii) **SSSK-IPE.KeyGen** $(msk, pp, \vec{y}) \longrightarrow sk_{\vec{y}}$: The key generation algorithm receives as input the master secret key msk, the public parameters pp, and a vector $\vec{y} \in \mathbb{Z}_q^n$ and outputs a secret key $sk_{\vec{y}}$.

(iv) **SSSK-IPE.Decrypt** $(pp, ct_{\vec{x}}, sk_{\vec{y}}) \longrightarrow m \in \mathbb{Z}_q$ or $\perp$: The decryption algorithm receives as input the public parameters pp, the ciphertext $ct_{\vec{x}}$, and a secret key $sk_{\vec{y}}$ and outputs either a value $m \in \mathbb{Z}_q$ or the dedicated symbol $\perp$.

We make the following correctness requirement: for all $(msk, pp) \xleftarrow{\$}$ **SSSK-IPE.Setup**$(1^\lambda, n)$, all $\vec{x}, \vec{y} \in \mathbb{Z}_q^n$, and for $ct_{\vec{x}} \xleftarrow{\$}$ **SSSK-IPE.Encrypt**$(msk, pp, \vec{x})$ and $sk_{\vec{y}} \xleftarrow{\$}$ **SSSK-IPE.KeyGen**$(msk, pp, \vec{y})$, **SSSK-IPE.Decrypt**$(pp, ct_{\vec{x}}, sk_{\vec{y}})$ is sure to output $\langle \vec{x}, \vec{y} \rangle$ whenever $\langle \vec{x}, \vec{y} \rangle \neq \perp$ with nonnegligible probability. The correctness requires the fact that it is $\langle \vec{x}, \vec{y} \rangle$ and not $\perp$ when $\langle \vec{x}, \vec{y} \rangle$ is from a fixed polynomial range of value inside $\mathbb{Z}_q$.

Definition 2 (indistinguishability-based security). A SSSK-IPE scheme has indistinguishability-based security if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the game defined as follows is negligible:

(1) A challenger runs **SSSK-IPE.Setup** to generate msk and pp, and pp is given to $\mathcal{A}$. It also chooses a random bit b.

(2) The challenger responds to the following two types of queries made by $\mathcal{A}$:

(i) Secret key query: $\mathcal{A}$ submits a pair of vectors $(\vec{y}_0, \vec{y}_1) \in \mathbb{Z}_q^n$ and the challenger com-

Experiment $\mathbf{Real}_{\mathcal{A}}^{\text{SSSK-IPE}}(1^\lambda)$:	**Experiment** $\mathbf{Ideal}_{\mathcal{A},S}^{\text{SSSK-IPE}}(1^\lambda)$:
$(\text{msk}, \text{pp}) \xleftarrow{\$} \text{SSSK-IPE.Setup}(1^\lambda, n)$	$(\text{msk}, st') \xleftarrow{\$} S(1^\lambda, n)$
$(\vec{x}, st) \xleftarrow{\$} \mathcal{A}_1^{\text{SSSK-IPE.KeyGen}(\text{msk},\cdot,\cdot)}(\text{pp})$	$(\vec{x}, st) \xleftarrow{\$} \mathcal{A}_1^{\text{SSSK-IPE.KeyGen}(\text{msk},\cdot,\cdot)}(\text{pp})$
$ct_{\vec{x}_i} \xleftarrow{\$} \text{SSSK-IPE.Encrypt}(\text{msk}, \text{pp}, \vec{x})$ for $i \in [\ell]$	$(\{ct_{\vec{x}_i}\}_{i\in[\ell]}) \xleftarrow{\$} S(\{\langle \vec{x}_i, \vec{y}_j \rangle\}_{i\in[\ell], j\in[p_1]})$ where $\vec{y}_1, \ldots, \vec{y}_{p_1}$ are key queries made by $\mathcal{A}_1$
$\alpha \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}(\text{msk},\cdot,\cdot)}(\{ct_{\vec{x}_i}\}_{i\in[\ell]}, st)$	$\alpha \xleftarrow{\$} \mathcal{A}_2^{\mathcal{O}'(\text{msk},st,\cdot)}(\{ct_{\vec{x}_i}\}_{i\in[\ell]}, st)$
output(α)	output(α)

Box 1: Real and ideal experiments.

putes and returns $sk_{\vec{y}} \longleftarrow$ **SSSK-IPE.KeyGen**$(msk, pp, \vec{y}_b)$.

(ii) Ciphertext query: $\mathcal{A}$ submits a pair of vectors $(\vec{x}_0, \vec{x}_1) \in \mathbb{Z}_q^n$ and the challenger computes and returns $ct_{\vec{x}} \longleftarrow$ **SSSK-IPE.Encrypt**$(msk, pp, \vec{x}_b)$.

(3) $\mathcal{A}$ outputs a bit b', and wins if $b' = b$.

A SSSK-IPE scheme is indistinguishability-based secure if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the above game, is $\text{Adv}_{\mathcal{A}}^{\text{SSSK-IPE}}(\lambda) = |\text{pr}[\text{Exp}_{\mathcal{A},\text{Game}_0}(1^\lambda) = 1] - \text{pr}[\text{Exp}_{\mathcal{A},\text{Game}_1}(1^\lambda) = 1]| < negl(\lambda)$, where this game is by definition Game_0 if $b = 0$ and Game_1 if $b = 1$.

Definition 3 (simulation-based security [1, 3, 7]). For a SSSK-IPE scheme, if there exits a PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ and a PPT simulator S, we define two experiments $\mathbf{Real}_{\mathcal{A}}^{\text{SSSK-IPE}}(1^\lambda)$ and $\mathbf{Ideal}_{\mathcal{A},S}^{\text{SSSK-IPE}}(1^\lambda)$ in Box 1. (in the public-key setting, the adversary $\mathcal{A}$ can encrypt plaintexts using the public parameters pp but $\mathcal{A}$ cannot in the secret key setting). Let ℓ be the number of challenge messages output by $\mathcal{A}_1$ and p_1 be the number of secret key queries in the first stage. The oracles $\mathcal{O}$ and $\mathcal{O}'$ are defined as follows:

(1) The oracle $\mathcal{O}(msk, \cdot) =$**SSSK-IPE.KeyGen**$(msk, \cdot, \cdot)$.

(2) The oracle $\mathcal{O}'(msk, st, \cdot)$ is the second stage of S, i.e., $S^{\langle \vec{x}_i, \vec{y}_j \rangle}(msk, st, \cdot)$ for $i \in [\ell], j \in [p_1]$, where $\vec{x}_i$ and $\vec{y}_j$ are inputs of the i^{th} ciphertext query and the jth secret key query by $\mathcal{A}_1$, respectively. The simulator S is stateful and after each query it updates the state st.

A SSSK-IPE scheme is simulation-based secure if there exists a PPT simulator S such that, for all PPT adversaries $\mathcal{A}$,

$$\mathbf{Real}_{\mathcal{A}}^{\text{SSSK-IPE}}\left(1^\lambda\right) \approx_c \mathbf{Ideal}_{\mathcal{A},S}^{\text{SSSK-IPE}}\left(1^\lambda\right). \quad (1)$$

2.2. Asymmetric Bilinear Pairing Groups

Definition 4 (asymmetric bilinear pairing groups). We say that an algorithm $\mathcal{G}_{abpg}(1^\lambda)$ is an asymmetric bilinear group generator and it outputs a bilinear pairing group which is defined by the tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, where q is a prime and $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ and are cyclic groups of order q, with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$ with the following properties:

(i) (Bilinearity) $\forall g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2, a, b \in \mathbb{Z}_q$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ and

(ii) (Nondegeneracy) $\exists g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2)$ has order q in $\mathbb{G}_T$.

2.3. External Decisional Linear Assumption

Definition 5 (external decisional linear (XDLIN) assumption [33]). $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is a tuple produced by $\mathcal{G}_{abpg}(1^\lambda)$. Consider the following problem: given the distributions $\mathcal{G}_b^{\text{XDLIN}}(1^\lambda) = ((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \xi g_1, \kappa g_1, \delta\xi g_1, \sigma\kappa g_1, \xi g_2, \kappa g_2, \delta\xi g_2, \sigma\kappa g_2, Y_b)$ for $b \in 0, 1$, where $\xi, \kappa, \delta, \sigma \xleftarrow{\$} \mathbb{Z}_q$, $Y_0 = (\delta + \sigma)g_c$, $Y_1 = (\delta + \sigma + \rho)g_c$, $\rho \xleftarrow{\$} \mathbb{Z}_q$ and $c \in \{0, 1\}$, output $\mathcal{G}_0^{\text{XDLIN}}$ if b is 0, and output $\mathcal{G}_1^{\text{XDLIN}}$ otherwise, we refer to the problem as the External Decisional Linear (XDLIN) problem.

For a PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ is defined as $\text{Adv}_{\mathcal{A}}^{\text{XDLIN}}(\lambda) = |\text{Pr}[\mathcal{A}(1^\lambda, \mathcal{G}_0^{\text{XDLIN}}) \longrightarrow 1] - \text{Pr}[\mathcal{A}(1^\lambda, \mathcal{G}_1^{\text{XDLIN}}(1^\lambda)) \longrightarrow 1]|$. If for all PPT algorithms $\mathcal{A}$, $\text{Adv}_{\mathcal{A}}^{\text{XDLIN}}(\lambda)$ is negligible in λ, we say that $\mathcal{G}_b^{\text{XDLIN}}(1^\lambda)$ satisfies the XDLIN assumption.

2.4. Dual Pairing Vector Spaces

Definition 6 (dual pairing vector spaces (DPVS) [31, 32]). A dual pairing vector space (DPVS) $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E)$ is directly defined by the tuple $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}_{abpg}(1^\lambda)$. $\mathbb{V} = \mathbb{G}_1^n$ and $\mathbb{V}^* = \mathbb{G}_2^n$ over $\mathbb{Z}_q^n$ are n-dimensional vector spaces. $\mathbb{A} = \{\vec{a}_1, \ldots, \vec{a}_n\}$ of $\mathbb{V}$ and $\mathbb{A}^* = \{\vec{a}_1^*, \ldots, \vec{a}_n^*\}$ of $\mathbb{V}^*$ are canonical bases, where $\vec{a}_i = (0^{i-1}, g_1, 0^{n-i})$ and $\vec{a}_i^* = (0^{i-1}, g_2, 0^{n-i})$. $E : \mathbb{V} \times \mathbb{V}^* \longrightarrow \mathbb{G}_T$ is pairing which is defined by $E(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^n e(X_i, Y_i) \in \mathbb{G}_T$, where $\boldsymbol{x} = (X_1, \ldots X_n) \in \mathbb{V}$ and $\boldsymbol{y} = (Y_1, \ldots Y_n) \in \mathbb{V}^*$ with the following properties:

(1) (Bilinearity) $E(a\vec{x}, b\vec{y}) = E(\vec{x}, \vec{y})^{ab}$ for $a, b \in \mathbb{Z}_q$ and

(2) (Nondegeneracy) if $E(\vec{x}, \vec{y}) = 1$ for all $\vec{y} \in \mathbb{V}^*$, then $\vec{x} = \vec{0}$.

Let $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E)$ be the output of algorithm $\mathcal{G}_{dpvs}(1^\lambda, n, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$, where $n \in \mathbb{N}$.

We then describe random dual orthonormal basis generator $\mathcal{G}_{ob}(1^\lambda, n)$ as follows:

$\mathcal{G}_{ob}(1^\lambda, n) : (q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E) \xleftarrow{\$} \mathcal{G}_{dpvs}(1^\lambda, n, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e))$, $\mathbf{B} = (\chi_{i,j}) \xleftarrow{\$} GL(n, \mathbb{Z}_q)$, $(\phi_{i,j}) = \psi(\mathbf{B}^T)^{-1}$,
$\overrightarrow{b}_i = \sum_{j=1}^n \chi_{i,j} \overrightarrow{a}_j$, $\mathbb{B} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n\}$,
$\overrightarrow{b}_i^* = \sum_{j=1}^n \phi_{i,j} \overrightarrow{a}_j^*$, $\mathbb{B}^* = \{\overrightarrow{b}_1^*, \ldots, \overrightarrow{b}_n^*\}$, $g_T = e(g_1, g_2)^\psi$,
returning $(\mathbb{B}, \mathbb{B}^*)$.

Let $(\overrightarrow{x})_{\mathbb{B}}$ denote $\sum_{i=1}^n x_i \overrightarrow{b}_i$, where $\overrightarrow{x} = (x_1, \ldots, x_n)^T \in \mathbb{Z}_q^n$ and $\mathbb{B} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n\}$. Then we have

$$\begin{aligned} E\left(\left(\overrightarrow{x}\right)_{\mathbb{A}}, \left(\overrightarrow{y}\right)_{\mathbb{A}^*}\right) &= \prod_{i=1}^n e\left(x_i g_1, y_i g_2\right) \\ &= e\left(g_1, g_2\right)^{\sum_{i=1}^n x_i y_i} \\ &= e\left(g_1, g_2\right)^{\langle \overrightarrow{x}, \overrightarrow{y} \rangle}, \\ E\left(\left(\overrightarrow{x}\right)_{\mathbb{B}}, \left(\overrightarrow{y}\right)_{\mathbb{B}^*}\right) &= E\left(\left(\mathbf{B}\overrightarrow{x}\right)_{\mathbb{A}}, \left(\psi\left(\mathbf{B}^T\right)^{-1}\overrightarrow{y}\right)_{\mathbb{A}^*}\right) \\ &= e\left(g_1, g_2\right)^{\psi \mathbf{B}\overrightarrow{x} \cdot (\mathbf{B}^T)^{-1}\overrightarrow{y}} = g_T^{\langle \overrightarrow{x}, \overrightarrow{y} \rangle}. \end{aligned} \tag{2}$$

3. Our SSSK-IPE Scheme

In this section, we present the construction of SSSK-IPE.

SSSK-IPE.Setup$(1^\lambda, \mathrm{n}) \longrightarrow$ (msk, pp): The setup algorithm runs $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}_{abpg}(1^\lambda)$. It then generates

$$\begin{aligned} &\left(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E\right) \\ &\quad \xleftarrow{\$} \mathcal{G}_{dpvs}\left(1^\lambda, n+6, \left(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e\right)\right), \\ &\left(\mathbb{B} = \left\{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_{n+6}\right\}, \mathbb{B}^* = \left\{\overrightarrow{b}_1^*, \ldots, \overrightarrow{b}_{n+6}^*\right\}\right) \\ &\quad \xleftarrow{\$} \mathcal{G}_{ob}\left(1^\lambda, n+6\right). \end{aligned} \tag{3}$$

The algorithm outputs msk= $(\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*)$, where $\widehat{\mathbb{B}} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n, \overrightarrow{b}_{n+1}, \overrightarrow{b}_{n+2}, \overrightarrow{b}_{n+5}\}$, $\widehat{\mathbb{B}}^* = \{\overrightarrow{b}_1^*, \ldots, \overrightarrow{b}_n^*, \overrightarrow{b}_{n+3}^*, \overrightarrow{b}_{n+4}^*, \overrightarrow{b}_{n+6}^*\}$, and pp $= (q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}_1, \mathbb{A}_1^*, E)$.

SSSK-IPE.Encrypt(msk, pp, $\overrightarrow{x}) \longrightarrow ct_{\overrightarrow{x}}$: The encryption algorithm samples $\alpha, \beta, \eta \xleftarrow{\$} \mathbb{Z}_q$ independently and uniformly at random and outputs

$$ct_{\overrightarrow{x}} = \left(\overrightarrow{x}, \alpha, \beta, 0, 0, \eta, 0\right)_{\mathbb{B}}. \tag{4}$$

SSSK-IPE.KeyGen(msk, pp, $\overrightarrow{y}) \longrightarrow sk_{\overrightarrow{y}}$: The secret key generation algorithm samples $\theta, \gamma, \zeta \xleftarrow{\$} \mathbb{Z}_q$ independently and uniformly at random and outputs

$$sk_{\overrightarrow{y}} = \left(\overrightarrow{y}, 0, 0, \theta, \gamma, 0, \zeta\right)_{\mathbb{B}^*}. \tag{5}$$

SSSK-IPE.Decrypt $(\mathrm{pp}, ct_{\overrightarrow{x}}, sk_{\overrightarrow{y}}) \longrightarrow m \in \mathbb{Z}_q$ or $\perp$: The decryption algorithm outputs

$$d = E\left(ct_{\overrightarrow{x}}, sk_{\overrightarrow{y}}\right). \tag{6}$$

It then attempts to determine $m \in \mathbb{Z}_q$ such that $g_T^m = d$. If there is m that satisfies the equation, the algorithm outputs m. Otherwise, it outputs $\perp$. Due to a polynomial-size range of possible values for m, the decryption algorithm certainly runs in polynomial time.

Correctness. For any $ct_{\overrightarrow{x}}$ and $sk_{\overrightarrow{y}}$ by calling **SSSK-IPE.Encrypt**(msk, pp, $\overrightarrow{x}$) and **SSSK-IPE.KeyGen**(msk, pp, $\overrightarrow{y}$), respectively, the pairing evaluations in the decryption algorithm proceed as follows:

$$d = E\left(ct_{\overrightarrow{x}}, sk_{\overrightarrow{y}}\right) = g_T^{\langle \overrightarrow{x}, \overrightarrow{y} \rangle}. \tag{7}$$

If the decryption algorithm takes polynomial time in the size of the plaintext space, it will output $m = \langle \overrightarrow{x}, \overrightarrow{y} \rangle$ as desired.

Remark 7. We can easily notice that our scheme is malleable, where a ciphertext can be created from certain other ciphertexts. The scheme in [18] is also malleable, while it seems difficult to prove the schemes in [16, 17, 19] to be malleable.

Remark 8. In contrast to the scheme of Tomida et al. [18], we provides a more efficient scheme with better security strength. The construction of [18] achieves indistinguishability-based security where the constraint on the adversaries is that, for all ciphertext queries $(\overrightarrow{x}_0, \overrightarrow{x}_1)$ and all key queries $(\overrightarrow{y}_0, \overrightarrow{y}_1)$, $\langle \overrightarrow{x}_0, \overrightarrow{y}_0 \rangle = \langle \overrightarrow{x}_1, \overrightarrow{y}_1 \rangle$ should hold. $\mathbb{B}$ and $\mathbb{B}^*$ of dimension $2n + 5$ are introduced in their construction. There are at least $2n$ dimensions of vector spaces because of changing a ciphertext including $\overrightarrow{x}_0$ (respectively, a secret key including $\overrightarrow{y}_0$) into one including $\overrightarrow{x}_1$ (respectively, a secret key including $\overrightarrow{y}_1$) in the security game. As mentioned in Section 1, in our scheme, a hidden dimension can be used to convert a corresponding coefficient to another coefficient in a secret key or a ciphertext, so that no PPT adversary can distinguish the two hybrid experiments. Our scheme therefore saves n dimensions of vector spaces.

4. Security Proof

4.1. Theorem 9. Now, we prove that our construction has many-AD-SIM (many messages secure, adaptive simulation definition) security from the XDLIN assumption in the standard model against adaptive adversaries obtaining an unbounded number of ciphertexts and secret keys.

Theorem 9. *Under the XDLIN assumption the proposed scheme is many-AD-SIM-secure.*

We will show Lemmas 15, 16, 17, 18, and 19 for the security proof of Theorem 9.

4.2. Lemmas

Definition 10 (problem 0). Problem 0 is to guess $b \in \{0,1\}$, given $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \mathbb{B}, \widehat{\mathbb{B}}^*, \overrightarrow{y}_b, \kappa g_1, \xi g_2)$, where

$(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e) \xleftarrow{\$} \mathcal{G}_{abpg}(1^\lambda)$,

$\mathbf{B} = (\chi_{i,j}) \xleftarrow{\$} GL(3, \mathbb{Z}_q)$ and $(\phi_{i,j}) = (\mathbf{B}^T)^{-1}$,

$\kappa \xleftarrow{\$} \mathbb{Z}_q, \overrightarrow{b}_i = \kappa \sum_{j=1}^n \chi_{i,j} \overrightarrow{a}_j$ for $i = 1, 2, 3$, $\mathbb{B} = \{\overrightarrow{b}_1, \overrightarrow{b}_2, \overrightarrow{b}_3\}$,

$\xi \xleftarrow{\$} \mathbb{Z}_q, \overrightarrow{b}^*_i = \xi \sum_{j=1}^n \phi_{i,j} \overrightarrow{a}^*_j$ for $i = 1, 3$, $\widehat{\mathbb{B}}^* = \{\overrightarrow{b}^*_1, \overrightarrow{b}^*_3\}$,

$g_T = e(g_1, g_2)^{\kappa\xi}$ and $\delta, \sigma \xleftarrow{\$} \mathbb{Z}_q, \rho \xleftarrow{\$} \mathbb{Z}_q^\times$, and

$\overrightarrow{y}_0 = (\delta, 0, \sigma)_{\mathbb{B}}$ and $\overrightarrow{y}_1 = (\delta, \rho, \sigma)_{\mathbb{B}}$.

Definition 11 (problem 1). Problem 1 is to guess $b \in \{0,1\}$, given $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \overrightarrow{Y}_b)$, where

$(\mathbb{B}, \mathbb{B}^*, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, n+6)$,

$\widehat{\mathbb{B}} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n, \overrightarrow{b}_{n+1}, \overrightarrow{b}_{n+2}, \overrightarrow{b}_{n+5}\}$,

$\widehat{\mathbb{B}}^* = \{\overrightarrow{b}^*_1, \ldots, \overrightarrow{b}^*_n, \overrightarrow{b}^*_{n+3}, \overrightarrow{b}^*_{n+4}\}$,

$\alpha, \beta \xleftarrow{\$} \mathbb{Z}_q, \eta \xleftarrow{\$} \mathbb{Z}_q^\times$, and

$\overrightarrow{Y}_0 = (0^n, \alpha, \beta, 0, 0, \eta, 0)_{\mathbb{B}}$ and $\overrightarrow{Y}_1 = (0^n, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{B}}$.

Definition 12 (problem 2). Problem 2 is to guess $b \in \{0,1\}$, given $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \overrightarrow{Y}_b)$, where

$(\mathbb{B}, \mathbb{B}^*, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, n+6)$,

$\widehat{\mathbb{B}} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n, \overrightarrow{b}_{n+1}, \overrightarrow{b}_{n+2}, \overrightarrow{b}_{n+6}\}$,

$\widehat{\mathbb{B}}^* = \{\overrightarrow{b}^*_1, \ldots, \overrightarrow{b}^*_n, \overrightarrow{b}^*_{n+3}, \overrightarrow{b}^*_{n+4}\}$,

$\alpha, \beta, \eta \xleftarrow{\$} \mathbb{Z}_q, \zeta' \xleftarrow{\$} \mathbb{Z}_q^\times$, and

$\overrightarrow{Y}_0 = (0^n, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{B}}$ and $\overrightarrow{Y}_1 = (0^n, \alpha, \beta, 0, 0, 0, \zeta')_{\mathbb{B}}$.

Definition 13 (problem 3). Problem 3 is to guess $b \in \{0,1\}$, given $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \overrightarrow{Y}^*_b)$, where

$(\mathbb{B}, \mathbb{B}^*, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, n+6)$,

$\widehat{\mathbb{B}} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n, \overrightarrow{b}_{n+1}, \overrightarrow{b}_{n+2}\}$,

$\widehat{\mathbb{B}}^* = \{\overrightarrow{b}^*_1, \ldots, \overrightarrow{b}^*_n, \overrightarrow{b}^*_{n+3}, \overrightarrow{b}^*_{n+4}, \overrightarrow{b}^*_{n+6}\}$,

$\theta, \gamma, \zeta \xleftarrow{\$} \mathbb{Z}_q, \eta' \xleftarrow{\$} \mathbb{Z}_q^\times$, and

$\overrightarrow{Y}^*_0 = (0^n, 0, 0, \theta, \gamma, 0, \zeta)_{\mathbb{B}^*}$ and $\overrightarrow{Y}^*_1 = (0^n, 0, 0, \theta, \gamma, \eta', \zeta)_{\mathbb{B}^*}$.

Definition 14 (problem 4). Problem 4 is to guess $b \in \{0,1\}$, given $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \overrightarrow{Y}^*_b)$, where

$(\mathbb{B}, \mathbb{B}^*, (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)) \xleftarrow{\$} \mathcal{G}_{ob}(1^\lambda, n+6)$,

$\widehat{\mathbb{B}} = \{\overrightarrow{b}_1, \ldots, \overrightarrow{b}_n, \overrightarrow{b}_{n+1}, \overrightarrow{b}_{n+2}\}$,

$\widehat{\mathbb{B}}^* = \{\overrightarrow{b}^*_1, \ldots, \overrightarrow{b}^*_n, \overrightarrow{b}^*_{n+3}, \overrightarrow{b}^*_{n+4}, \overrightarrow{b}^*_{n+5}\}$,

$\theta, \gamma, \zeta \xleftarrow{\$} \mathbb{Z}_q, \eta' \xleftarrow{\$} \mathbb{Z}_q^\times$, and

$\overrightarrow{Y}^*_0 = (0^n, 0, 0, \theta, \gamma, \eta', \zeta)_{\mathbb{B}^*}$ and $\overrightarrow{Y}^*_1 = (0^n, 0, 0, \theta, \gamma, \eta', 0)_{\mathbb{B}^*}$.

For a PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ for Problem n, where $n = 0, 1, 2, 3, 4$, is defined as

$$\begin{aligned} &\mathrm{Adv}^{\mathrm{Prob_n}}_{\mathcal{A}}(\lambda) \\ &= \left|\Pr\left[\mathrm{Exp}^{P_0}_{\mathcal{A}}\left(1^\lambda\right) = 1\right] - \Pr\left[\mathrm{Exp}^{P_1}_{\mathcal{A}}\left(1^\lambda\right) = 1\right]\right|, \end{aligned} \tag{8}$$

where the instance is by definition P_0 if $b = 0$ and P_1 if $b = 1$.

Lemma 15. *Suppose that the XDLIN assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$. Then for all PPT adversary $\mathcal{B}$, there is an adversary $\mathcal{A}$ such that $Adv^{Prob_0}_{\mathcal{B}}(\lambda) \le Adv^{XDLIN}_{\mathcal{A}}(\lambda) + 5/q$.*

Proof. Let W be a random linear transformation. On inputting an instance of XDLIN $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \xi g_1, \kappa g_1, \delta\xi g_1, \sigma\kappa g_1, \xi g_2, \kappa g_2, \delta\xi g_2, \sigma\kappa g_2, Y_b)$, $\mathcal{A}$ sets

$g_T = e(\kappa\mathbb{G}_1, \xi\mathbb{G}_2)$, $\overrightarrow{u}_1 = (\xi, 0, 1)_{\mathbb{A}}$, and $\overrightarrow{u}_2 = (0, 0, 1)_{\mathbb{A}}$,

$\overrightarrow{u}_3 = (0, \kappa, 1)_{\mathbb{A}}, \overrightarrow{u}^*_1 = (\kappa, 0, 0)_{\mathbb{A}^*}$, and $\overrightarrow{u}^*_2 = (-\kappa, -\xi, \kappa\xi)_{\mathbb{A}^*}$,

$\overrightarrow{u}^*_3 = (0, \xi, 0)_{\mathbb{A}^*}$ and $\overrightarrow{w}_b = (\delta\xi\mathbb{G}_1, \sigma\kappa\mathbb{G}_1, Y_b)$.

$\mathcal{A}$ can calculate $\overrightarrow{u}_i$ for $i = 1, 2, 3$ and $\overrightarrow{u}^*_i$ for $i = 1, 3$. Next, $\mathcal{A}$ generates $\overrightarrow{b}_i = W(\overrightarrow{u}_i)$ for $i = 1, 2, 3$, $\overrightarrow{b}^*_i = (W^{-1})^T(\overrightarrow{u}^*_i)$ for $i = 1, 2, 3$, and $\overrightarrow{y}_b = W(\overrightarrow{w}_b)$. Then $\mathcal{A}$ sets $\mathbb{B} = (\overrightarrow{b}_1, \overrightarrow{b}_2, \overrightarrow{b}_3)$, $\widehat{\mathbb{B}}^* = (\overrightarrow{b}^*_1, \overrightarrow{b}^*_3)$. Finally, $\mathcal{A}$ gives $\mathcal{B}((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \mathbb{B}, \widehat{\mathbb{B}}^*, \overrightarrow{y}_b, \kappa g_1, \xi g_2)$.

If $b = 0$ and $Y_b = Y_0 = (\delta + \sigma)\mathbb{G}_1$, then $\overrightarrow{w}_0 = (\delta\xi\mathbb{G}_1, \sigma\kappa\mathbb{G}_1, (\delta + \sigma)\mathbb{G}_1) = (\delta\xi, \sigma\kappa, (\delta + \sigma))_{\mathbb{A}} = \delta\overrightarrow{u}_1 + \sigma\overrightarrow{u}_2$ and $y_0 = W(\overrightarrow{w}_0) = W(\delta\overrightarrow{u}_1 + \sigma\overrightarrow{u}_2) = (\delta, 0, \sigma)_{\mathbb{B}}$ when $\kappa \neq 0$ and $\xi \neq 0$ (with probability $2/q$).

If $b = 1$ and $Y_b = Y_1 = (\delta + \rho + \sigma)\mathbb{G}_1$, then $\overrightarrow{w}_1 = (\delta\xi\mathbb{G}_1, \sigma\kappa\mathbb{G}_1, (\delta + \rho + \sigma)\mathbb{G}_1) = (\delta\xi, \sigma\kappa, (\delta + \rho + \sigma))_{\mathbb{A}} = \delta\overrightarrow{u}_1 + \rho\overrightarrow{u}_2 + \sigma\overrightarrow{u}_3$ and $y_1 = W(\overrightarrow{w}_1) = W(\delta\overrightarrow{u}_1 + \rho\overrightarrow{u}_2 + \sigma\overrightarrow{u}_3) = (\delta, \rho, \sigma)_{\mathbb{B}}$ when $\kappa \neq 0, \xi \neq 0$ and $\rho \neq 0$ (with probability $3/q$). Therefore, $\mathrm{Adv}^{\mathrm{Prob_0}}_{\mathcal{B}}(\lambda) \le \mathrm{Adv}^{\mathrm{XDLIN}}_{\mathcal{A}}(\lambda) + 5/q$. □

Lemma 16. *Suppose that the XDLIN assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$. Then for all PPT adversary $\mathcal{B}$, there is an adversary $\mathcal{A}$ such that $Adv^{Prob_1}_{\mathcal{B}}(\lambda) \le Adv^{Prob_0}_{\mathcal{A}}(\lambda)$.*

Proof. Let W be a random linear transformation. On inputting an instance of Porblem 0 $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \mathbb{B}, \widehat{\mathbb{B}}^*, \overrightarrow{y}_b, \kappa g_1, \xi g_2)$, $\mathscr{A}$ calculates

$$h_i = W(0^{i+2}, \kappa g_1, 0^{n+3-i}) \text{ for } i = 1, \ldots, n,$$

$$h_i = W(0^{i}, \kappa g_1, 0^{n+5-i}) \text{ for } i = n+3, n+4, n+5,$$

$$h_{n+1} = W(\overrightarrow{b}_1, 0^{n+3}), h_{n+2} = W(\overrightarrow{b}_3, 0^{n+3}),$$

$$h_{n+6} = W(\overrightarrow{b}_2, 0^{n+3}),$$

$$h_i^* = (W^{-1})^T(0^{i+2}, \xi g_2, 0^{n+3-i}) \text{ for } i = 1, \ldots, n,$$

$$h_i^* = (W^{-1})^T(0^{i}, \xi g_2, 0^{n+5-i}) \text{ for } i = n+3, n+4, n+5,$$

$$h_{n+1}^* = (W^{-1})^T(\overrightarrow{b}_1^*, 0^{n+3}), h_{n+2}^* = (W^{-1})^T(\overrightarrow{b}_3^*, 0^{n+3}),$$

$$h_{n+6}^* = (W^{-1})^T(\overrightarrow{b}_2^*, 0^{n+3}) \text{ and } \overrightarrow{Y}_b = W(\overrightarrow{y}_b, 0^{n+3}),$$

$$\mathbb{H} = (\overrightarrow{h}_1, \ldots, \overrightarrow{h}_{n+6}) \text{ and } \mathbb{H}^* = (\overrightarrow{h}_1^*, \ldots, \overrightarrow{h}_{n+6}^*).$$

Then $\mathscr{A}$ sets $\widehat{\mathbb{H}} = (\overrightarrow{h}_1, \ldots, \overrightarrow{h}_n, \overrightarrow{h}_{n+1}, \overrightarrow{h}_{n+2}, \overrightarrow{h}_{n+5})$ and $\widehat{\mathbb{H}}^* = (\overrightarrow{h}_1^*, \ldots, \overrightarrow{h}_n^*, \overrightarrow{h}_{n+3}^*, \overrightarrow{h}_{n+4}^*)$. Finally, $\mathscr{A}$ gives $\mathscr{B}((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{H}}, \widehat{\mathbb{H}}^*, \overrightarrow{Y}_b)$. If $b = 0$, $\overrightarrow{Y}_0 = (0^n, \alpha, \beta, 0, 0, \eta, 0)_{\mathbb{H}}$, and if $b = 1$, $\overrightarrow{Y}_1 = (0^n, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{H}}$), where $\delta = \alpha$, $\rho = \zeta'$, and $\sigma = \beta$. This is the same as an instance of Problem 0. □

Lemma 17. *Suppose that the XDLIN assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$. Then for all PPT adversary $\mathscr{B}$, there is an adversary $\mathscr{A}$ such that $Adv_{\mathscr{B}}^{Prob_2}(\lambda) \leq Adv_{\mathscr{A}}^{Prob_0}(\lambda)$.*

Proof. Let W be a random linear transformation. On inputting an instance of Porblem 0 $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \mathbb{B}, \widehat{\mathbb{B}}^*, \overrightarrow{y}_b, \kappa g_1, \xi g_2)$, $\mathscr{A}$ calculates

$$h_i = W(0^{i+2}, \kappa g_1, 0^{n+3-i}) \text{ for } i = 1, ..., n,$$

$$h_i = W(0^{i}, \kappa g_1, 0^{n+5-i}) \text{ for } i = n+3, n+4, n+6,$$

$$h_{n+1} = W(\overrightarrow{b}_1, 0^{n+3}) \text{ and } h_{n+2} = W(\overrightarrow{b}_3, 0^{n+3}),$$

$$h_{n+5} = W(\overrightarrow{b}_2, 0^{n+3}),$$

$$h_i^* = (W^{-1})^T(0^{i+2}, \xi g_2, 0^{n+3-i}) \text{ for } i = 1, \ldots, n,$$

$$h_i^* = (W^{-1})^T(0^{i}, \xi g_2, 0^{n+5-i}) \text{ for } i = n+3, n+4, n+6,$$

$$h_{n+1}^* = (W^{-1})^T(\overrightarrow{b}_1^*, 0^{n+3}) \text{ and } h_{n+2}^* = (W^{-1})^T(\overrightarrow{b}_3^*, 0^{n+3}),$$

$$h_{n+5}^* = (W^{-1})^T(\overrightarrow{b}_2^*, 0^{n+3}) \text{ and } \overrightarrow{Y}_b = W(\overrightarrow{y}_b, 0^{n+3}),$$

$$\mathbb{H} = (\overrightarrow{h}_1, \ldots, \overrightarrow{h}_{n+6}), \text{ and } \mathbb{H}^* = (\overrightarrow{h}_1^*, \ldots, \overrightarrow{h}_{n+6}^*).$$

Then $\mathscr{A}$ sets $\widehat{\mathbb{H}} = (\overrightarrow{h}_1, \ldots, \overrightarrow{h}_n, \overrightarrow{h}_{n+1}, \overrightarrow{h}_{n+2}, \overrightarrow{h}_{n+6})$ and $\widehat{\mathbb{H}}^* = (\overrightarrow{h}_1^*, \ldots, \overrightarrow{h}_n^*, \overrightarrow{h}_{n+3}^*, \overrightarrow{h}_{n+4}^*)$. Finally, $\mathscr{A}$ gives $\mathscr{B}((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{H}}, \widehat{\mathbb{H}}^*, \overrightarrow{Y}_b)$. If $b = 0$, $\overrightarrow{Y}_0 = (0^n, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{H}}$, and if $b = 1$, $\overrightarrow{Y}_1 = (0^n, \alpha, \beta, 0, 0, 0, \zeta')_{\mathbb{H}}$, where $\delta = \alpha$, $\rho = \eta$, and $\sigma = \beta$. This is the same as an instance of Problem 0. □

Lemma 18. *Suppose that the XDLIN assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$. Then for all PPT adversary $\mathscr{B}$, there is an adversary $\mathscr{A}$ such that $Adv_{\mathscr{B}}^{Prob_3}(\lambda) \leq Adv_{\mathscr{A}}^{Prob_0}(\lambda)$.*

Proof. The proof follows in the same manner as Lemma 16. □

Lemma 19. *Suppose that the XDLIN assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$. Then for all PPT adversary $\mathscr{B}$, there is an adversary $\mathscr{A}$ such that $Adv_{\mathscr{B}}^{Prob_4}(\lambda) \leq Adv_{\mathscr{A}}^{Prob_0}(\lambda)$.*

Proof. The proof follows in the same manner as Lemma 17. □

4.3. Security Proof of Theorem 9. For the security proof, we follow the simulation-based security notion [1, 3]. A simulator responds to queries by an adversary $\mathscr{A}$ and provides simulated secret keys and simulated ciphertexts to $\mathscr{A}$. The simulator is made of three algorithms: $\widetilde{\textbf{Setup}}$, $\widetilde{\textbf{Encrypt}}$, and $\widetilde{\textbf{KeyGen}}$.

$\widetilde{\textbf{Setup}}$: It generates a master secret key msk and public parameters pp, which is transferred to $\mathscr{A}$. Specially, on input $(1^\lambda, n)$, it sets (msk, pp)⟵**SSSK-IPE.Setup**. The simulator will use the master secret key and the public parameters to answer the queries of $\mathscr{A}$ in $\widetilde{\textbf{Encrypt}}$ and $\widetilde{\textbf{KeyGen}}$.

$\widetilde{\textbf{Encrypt}}$: It simulates the ciphertexts of challenge messages $\overrightarrow{x}_1, \ldots, \overrightarrow{x}_\ell$, where $\overrightarrow{x}_1, \ldots, \overrightarrow{x}_\ell$ are output by $\mathscr{A}$ and ℓ is the number of the challenge messages. Let p_1 be the number of secret key queries in the first stage. $\widetilde{\textbf{Encrypt}}$ receives as input msk, pp, nonadaptive secret key queries $\overrightarrow{y}_1, \ldots, \overrightarrow{y}_{p_1}$ made by $\mathscr{A}$, together with $(\langle \overrightarrow{x}_i, \overrightarrow{y}_1 \rangle, \ldots, \langle \overrightarrow{x}_i, \overrightarrow{y}_{p_1} \rangle)$ for each $1 \leq i \leq \ell$, and the secret keys $(\overrightarrow{y}_1, sk_{\overrightarrow{y}_1}), \ldots, (\overrightarrow{y}_{p_1}, sk_{\overrightarrow{y}_{p_1}})$. The normal ciphertext is $(\overrightarrow{x}, \alpha, \beta, 0, 0, \eta, 0)_{\mathbb{B}}$ generated by **SSSK-IPE.Encrypt**, where $\alpha, \beta, \eta \xleftarrow{\$} \mathbb{Z}_q$. The simulated ciphertext is $(\overrightarrow{x}, \alpha, \beta, 0, 0, 0, \zeta')_{\mathbb{B}}$ generated by $\widetilde{\textbf{Encrypt}}$, where $\zeta' \xleftarrow{\$} \mathbb{Z}_q^\times$. In order to prove the views of $\mathscr{A}$ in **SSSK-IPE.Encrypt** and that in $\widetilde{\textbf{Encrypt}}$ have the same distribution, we define a new algorithm $\widetilde{\textbf{Encrypt}}'$, where $ct_{\overrightarrow{x}} = (\overrightarrow{x}, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{B}}$.

$\widetilde{\textbf{KeyGen}}$: It simulates the answer to the second stage queries of $\mathscr{A}$. It receives as input msk, pp, the vector $\overrightarrow{y}$, where $\overrightarrow{y}$ is the secret key query made by $\mathscr{A}$, and the values $\langle \overrightarrow{x}_1, \overrightarrow{y} \rangle, \ldots, \langle \overrightarrow{x}_\ell, \overrightarrow{y} \rangle$, where $\overrightarrow{x}_1, \ldots, \overrightarrow{x}_\ell$ are the challenge messages. The normal secret key is $(\overrightarrow{y}, 0, 0, \theta, \gamma, 0, \zeta)_{\mathbb{B}^*}$ generated by **SSSK-IPE.KeyGen**, where $\theta, \gamma, \zeta \xleftarrow{\$} \mathbb{Z}_q$. The simulated secret key is $(\overrightarrow{y}, 0, 0, \theta, \gamma, \eta', 0)_{\mathbb{B}^*}$ generated by $\widetilde{\textbf{KeyGen}}$, where $\eta' \xleftarrow{\$} \mathbb{Z}_q^\times$. Analogous to $\widetilde{\textbf{Encrypt}}$, we also define a new algorithm $\widetilde{\textbf{KeyGen}}'$ where $sk_{\overrightarrow{y}} = (\overrightarrow{y}, 0, 0, \theta, \gamma, \eta', \zeta)_{\mathbb{B}^*}$.

Next, we will prove it is indistinguishable between the output of an ideal world experiment and the output of the real world experiment via a hybrid argument.

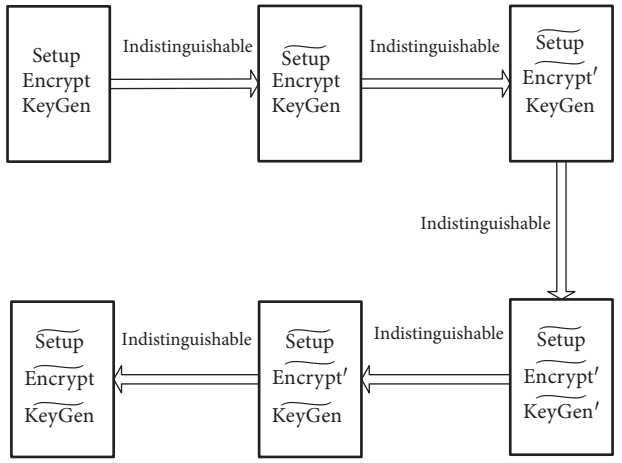

Figure 1: An overview of the security proof.

Proof. A high-level overview of the security proof is given in Figure 1. By a standard hybrid argument, we prove the distributions of the outputs of $\widetilde{\textbf{Encrypt}}$ and $\widetilde{\textbf{KeyGen}}$ are computationally indistinguishable from that of the normal ciphertexts and the normal secret keys, respectively. We proceed via a series of hybrid experiments $\mathbf{H}_0, \ldots, \mathbf{H}_5$, where $\mathbf{H}_0$ is the real world experiment and $\mathbf{H}_5$ is the ideal world experiment. We then prove that each hybrid experiment is indistinguishable from the neighboring one.

(i) **Hybrid** $\mathbf{H}_0$: This is the real experiment.

(ii) **Hybrid** $\mathbf{H}_1$: This experiment is the same as $\mathbf{H}_0$ except that the master secret key and the public parameters are generated by $\widetilde{\textbf{Setup}}$.

(iii) **Hybrid** $\mathbf{H}_2$: This experiment is the same as $\mathbf{H}_1$ except that every challenge ciphertext is $ct_{\vec{x}} = (\vec{x}, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{B}}$ which is generated by $\widetilde{\textbf{Encrypt}}'$.

(iv) **Hybrid** $\mathbf{H}_3$: This experiment is the same as $\mathbf{H}_2$ except that, for every key query $\vec{y}$, the corresponding secret key is $sk_{\vec{y}} = (\vec{y}, 0, 0, \theta, \gamma, \eta', \zeta)_{\mathbb{B}^*}$ which is generated by $\widetilde{\textbf{KeyGen}}'$.

(v) **Hybrid** $\mathbf{H}_4$: This experiment is the same as $\mathbf{H}_3$ except that, for every key query $\vec{y}$, the corresponding secret key is $sk_{\vec{y}} = (\vec{y}, 0, 0, \theta, \gamma, \eta', 0)_{\mathbb{B}^*}$ which is generated by $\widetilde{\textbf{KeyGen}}$.

(vi) **Hybrid** $\mathbf{H}_5$: This experiment is the same as $\mathbf{H}_4$ except that every challenge ciphertext is $ct_{\vec{x}} = (\vec{x}, \alpha, \beta, 0, 0, 0, \zeta')_{\mathbb{B}}$ which is generated by $\widetilde{\textbf{Encrypt}}$. □

Lemma 20. *For all PPT adversaries* $\mathcal{A}$, $\mathbf{H}_0 \approx_c \mathbf{H}_1$.

Proof. Because the master secret key and the public parameters are all generated by **SSSK-IPE.Setup** in $\mathbf{H}_0$ and $\mathbf{H}_1$, the view of $\mathcal{A}$ in $\mathbf{H}_0$ and that in $\mathbf{H}_1$ has the same distribution. □

Lemma 21. *Assuming that Problem 1 holds, for all PPT adversaries* $\mathcal{A}$, $\mathbf{H}_1 \approx_c \mathbf{H}_2$.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish the output distributions of $\mathbf{H}_1$ and $\mathbf{H}_2$. Then, we construct a PPT algorithm $\mathcal{B}$ which is given an instance of Problem 1 $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \vec{Y}_b)$ for $b \in \{0, 1\}$ and simulates $\mathbf{H}_1$ and $\mathbf{H}_2$.

Setup: $\mathcal{B}$ runs **SSSK-IPE.Setup**$(1^\lambda, \mathrm{n})$ and outputs msk= $(\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*)$ and pp= $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E)$. $\mathcal{B}$ gives $\mathcal{A}$ the public parameters pp and the master secret key msk is only known to $\mathcal{B}$.

Secret Key Queries: To answer the key queries made by $\mathcal{A}$, $\mathcal{B}$ runs algorithm **SSSK-IPE.Encrypt** to respond with $sk_{\vec{y}} = (\vec{y}, 0, 0, \theta, \gamma, 0, \zeta)_{\mathbb{B}^*}$.

Simulated Ciphertexts: $\mathcal{B}$ randomly chooses $k = \{1, \ldots, \ell\}$, where ℓ is the number of the ciphertext queries asked by the adversary $\mathcal{A}$. To answer the ith ciphertext query that $\mathcal{A}$ makes, $\mathcal{B}$ chooses $\alpha, \beta, \eta \xleftarrow{\$} \mathbb{Z}_q$ and $\zeta' \xleftarrow{\$} \mathbb{Z}_q^\times$ and computes and answers as

$$ct_{\vec{x}} = \sum_{i=1}^{n} x_i \vec{b}_i + \alpha \vec{b}_{n+1} + \beta \vec{b}_{n+2} + \eta \vec{b}_{n+5} + \zeta' \vec{b}_{n+6} \text{ if } i < k,$$

$$ct_{\vec{x}} = \sum_{i=1}^{n} x_i \vec{b}_i + \vec{Y}_b \text{ if } i = k,$$

$$ct_{\vec{x}} = \sum_{i=1}^{n} x_i \vec{b}_i + \alpha \vec{b}_{n+1} + \beta \vec{b}_{n+2} + \eta \vec{b}_{n+5} \text{ if } i > k.$$

The view of $\mathcal{A}$ is composed of the public parameters and the answers of the secret key queries and the ciphertext query. The public parameters in $\mathbf{H}_1$ and $\mathbf{H}_2$ are all generated by **SSSK-IPE.Setup** and thus have the same distribution, similar to the answers to the secret key queries. As for the answer to the ith ciphertext query, if $b = 0$ then the answer is distributed as in $\mathbf{H}_1$ and if $b = 1$ then the answer is distributed as in $\mathbf{H}_2$. □

Lemma 22. *Assuming that Problem 3 holds, for all PPT adversaries* $\mathcal{A}$, $\mathbf{H}_2 \approx_c \mathbf{H}_3$.

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ that can distinguish the output distributions of $\mathbf{H}_2$ and $\mathbf{H}_3$. Then, we construct a PPT algorithm $\mathcal{B}$ which is given an instance of Problem 3 $((q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e), \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \vec{Y}_b^*)$ for $b \in \{0, 1\}$ and simulates $\mathbf{H}_2$ and $\mathbf{H}_3$.

Setup: $\mathcal{B}$ runs **SSSK-IPE.Setup**$(1^\lambda, \mathrm{n})$ and outputs msk= $(\widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*)$ and pp= $(q, \mathbb{V}, \mathbb{V}^*, \mathbb{G}_T, \mathbb{A}, \mathbb{A}^*, E)$. $\mathcal{B}$ gives $\mathcal{A}$ the public parameters pp, and the master secret key msk is only known to $\mathcal{B}$.

First stage key queries: To answer every key query made by $\mathcal{A}$, $\mathcal{B}$ chooses random $\theta, \gamma, \zeta \xleftarrow{\$} \mathbb{Z}_q$, runs **SSSK-IPE.KeyGen** (msk, pp, $\vec{y}$), and responds with $sk_{\vec{y}} = (\vec{y}, 0, 0, \theta, \gamma, 0, \zeta)_{\mathbb{B}^*}$.

Simulated ciphertexts: To answer every ciphertext query that $\mathcal{A}$ makes, $\mathcal{B}$ chooses random $\alpha, \beta, \eta \xleftarrow{\$} \mathbb{Z}_q$ and $\zeta' \xleftarrow{\$} \mathbb{Z}_q^\times$, runs $\widetilde{\textbf{Encrypt}}'$, and answers as $ct_{\vec{x}} = (\vec{x}, \alpha, \beta, 0, 0, \eta, \zeta')_{\mathbb{B}}$.

Second stage key queries: $\mathcal{B}$ randomly chooses $k = \{1, \ldots, p_2\}$, where p_2 is the number of the ciphertext queries asked by the adversary $\mathcal{A}$ in the second stage. To answer the

Table 2: Performance of 80 bit security level.

N	Encrypt	KeyGen	Decrypt	$\lvert sk_{\vec{y}}\rvert$
10	1.5ms	10.6ms	60.6ms	2.2KB
30	3.7ms	23.8ms	130.7ms	4.8KB
50	7.0ms	39.7ms	202.4ms	7.3KB
100	15.8ms	73.5ms	374.7ms	13.8KB
300	92.9ms	257.1ms	1.1s	39.6KB
500	242.7ms	516.4ms	1.8s	65.3KB
750	506.4ms	939.4ms	2.6s	97.5KB
1000	891.3ms	1.5s	3.5s	129.8KB

Table 3: Performance of 112 bit security level.

N	Encrypt	KeyGen	Decrypt	$\lvert sk_{\vec{y}}\rvert$
10	3.3ms	21.5ms	109.9ms	3.0KB
30	7.0ms	43.8ms	232.5ms	6.5KB
50	11.2ms	66.3ms	344.4ms	10.0KB
100	23.2ms	123.1ms	638.5ms	18.8KB
300	114.3ms	391.3ms	1.8s	54.0KB
500	269.9ms	728.8ms	2.9s	89.1KB
750	561.5ms	1.2s	4.4s	133.0KB
1000	964.6ms	1.9s	5.9s	177.0KB

ith key query that $\mathcal{A}$ makes, $\mathcal{B}$ chooses random $\theta, \gamma, \zeta \xleftarrow{\$} \mathbb{Z}_q$ and $\eta' \xleftarrow{\$} \mathbb{Z}_q^{\times}$ and computes and answers as

$$sk_{\vec{y}} = \sum_{i=1}^{n} y_i \vec{b}_i^{*} + \theta \vec{b}_{n+3}^{*} + \gamma \vec{b}_{n+4}^{*} + \eta' \vec{b}_{n+5}^{*} + \zeta \vec{b}_{n+6}^{*} \text{ if } i < k,$$

$$sk_{\vec{y}} = \sum_{i=1}^{n} y_i \vec{b}_i^{*} + \vec{Y}_b \text{ if } i = k,$$

$$sk_{\vec{y}} = \sum_{i=1}^{n} y_i \vec{b}_i^{*} + \theta \vec{b}_{n+3}^{*} + \gamma \vec{b}_{n+4}^{*} + \zeta \vec{b}_{n+6}^{*} \text{ if } i > k.$$

We observe the view of $\mathcal{A}$ in its interaction with $\mathcal{B}$. The public parameters in $\mathbf{H}_2$ and $\mathbf{H}_3$ are all generated by **SSSK-IPE.Setup** and thus have the same distribution, similarly for the answers to every ciphertext query where $ct_{\vec{x}}$ in $\mathbf{H}_2$ and $\mathbf{H}_3$ are all generated by $\widetilde{\textbf{Encrypt}}'$. As for the answer to the ith key query, if $b = 0$ then the answer is distributed as in $\mathbf{H}_2$ and if $b = 1$ then the answer is distributed as in $\mathbf{H}_3$. □

Lemma 23. *Assuming that Problem 4 holds, for all PPT adversaries $\mathcal{A}$, $\mathbf{H}_3 \approx_c \mathbf{H}_4$.*

Proof. The proof follows in the same manner as Lemma 22. □

Lemma 24. *Assuming that Problem 2 holds, for all PPT adversaries $\mathcal{A}$, $\mathbf{H}_4 \approx_c \mathbf{H}_5$.*

Proof. The proof follows in the same manner as Lemma 21. □

5. Evaluation

To assess the practicality of the SSSK-IPE scheme in Section 3, we implement it similar to FHIPE [25], where two libraries are used: the Charm [34] for the pairing group operations and FLINT [35] for the finite field arithmetic in $\mathbb{Z}_q$, and the algorithm in [36] is used for evaluating discrete logs. Then we make a comparison among the SSSK-IPE scheme and the previous schemes in Table 1 on the performance. The experiments are performed on a personal computer with a 3.4G 8-core processor and 4GB RAM memory.

5.1. Microbenchmarks. We denote N as the length of the binary vectors $\vec{x}, \vec{y}$ in the SSSK-IPE scheme. (We take binary vectors for the microbenchmarks, because they can make our search space small during decryption. TThis is similar to PHIPE in [25].). In the microbenchmarks (Tables 2 and 3), we evaluate the run times of **Encrypt**, **KeyGen,** and **Decrypt,** respectively, and the ciphertext sizes $\lvert sk_{\vec{y}}\rvert$ which change with the values of N. Both 80 bit security level and 112 bit security level in which pairing curves are MNT159 and MNT224, respectively, are considered, respectively. In the offline stage, the algorithm **Setup** computes the inverse of $\mathbb{B}^T$, naive computation complexity of which is $O(n^3)$, in order to generate $\mathbb{B}^*$ from $\mathbb{B}$. Note that we consider an amortized complexity model, in which a larger amount of computational work is invested in the offline phase for obtaining efficiency during the online phase. Although **Encrypt** and **KeyGen** have the same number of group exponentiations over $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, there is a difference in the run times between them. This is because the artificial distinction of

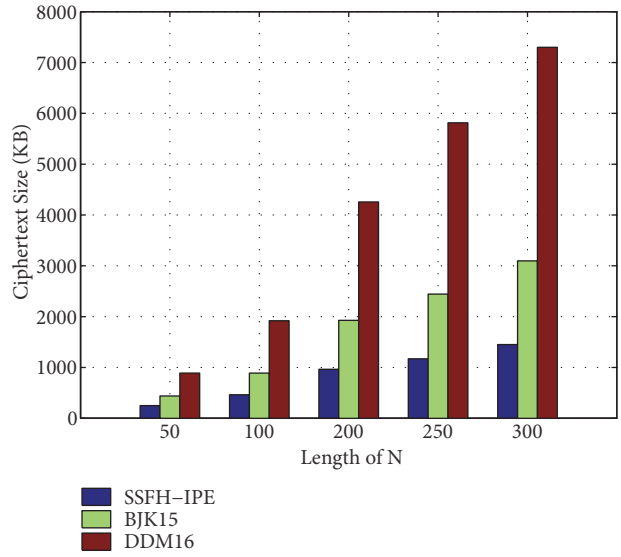

Figure 2: Run times relative to prior schemes varying with vector length N (80 bit security level).

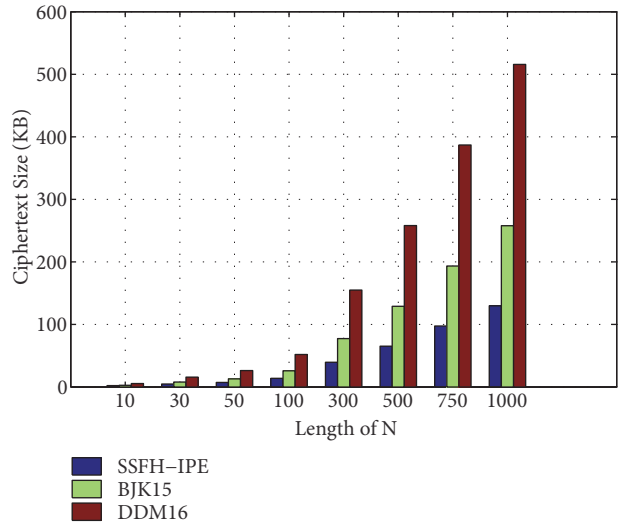

Figure 3: Ciphertext sizes relative to prior schemes varying with vector length N (80 bit security level).

implementing asymmetric pairing groups. The run time of **Decrypt** is more than that of **Encrypt** or **KeyGen**, as a pairing operation consumes more time than a group exponentiation.

5.2. Comparison with Previous Schemes. In Figures 2 and 3, we compare our scheme with related works in Table 1. Of all the three schemes BJK15[16], TAO16 [18], and ZZL17 [19] in Table 1, BJK15 is selected as the representative of comparison, because the numbers of both group exponentiations and pairing operations of them are almost equal. Figure 2 shows that the run times of the schemes increase with N ranging from 50 to 300. This is due to the increase in vector length N which results in more group exponentiations and more pairing operations. Compared with BJK15 and DDM16, the SSSK-IPE scheme is 1.8-2.1 times faster and 3.6-5 times faster, respectively. Figure 3 shows that the ciphertext sizes of the schemes increase with vector length N. Compared with BJK15 and DDM16, the ciphertext size of the SSSK-IPE scheme is 1.2 - 2.1 times less and 2.5 - 4 times less, respectively.

6. Applications

In this section, we show some direct applications of secret key IPE scheme.

6.1. Search on Encrypted Data. An interesting application of secret key IPE is to achieve encrypted data searching. Consider that a company uses a cloud to back up a large number of files. The company wishes to protect the privacy of these files by encrypting vectors representation of them prior to uploading them to the cloud. The company provides the master secret key for the secret key IPE scheme to an authorized employee who can search encrypted files in the cloud. Later on, the employee wants to search files corresponding to respective function. Firstly he transitions his query into a vector, computes the secret key of it, and sends the secret key to the cloud. The cloud then builds searches on disjunction queries, conjunction queries, and arbitrary CNF and DNF formulas and returns those files which satisfy queries, as shown in Figure 4.

Specifically, to encrypt a keyword value a in the finite field $\mathbb{Z}_q$, we can encode the value into a vector $\overrightarrow{x} = (1, a, \ldots, a^{n-1})$. Assume a secret key is created based on a disjunction query "a_1 OR a_2 Or a_3". It can be considered as the univariate polynomial $p(x) = (x-a_1)(x-a_2)(x-a_3)$ which is expanded in standard form as $p(x) = c_0 + c_1x + c_2x^2 + c_3x^3$, where c_i are the corresponding coefficients. The polynomial $p(x)$ is encoded as a vector $\overrightarrow{y} = (c_0, c_1, c_2, c_3, 0, \ldots, 0)$. Note that $\langle \overrightarrow{x}, \overrightarrow{y} \rangle = c_0 + c_1a + c_2a^2 + c_3a^3$. If $a = a_1$ or $a = a_2$ or $a = a_3$ such that $p(x) = 0$, the secret key will match the ciphertext for the value a. However, we notice that expressiveness of IPE is less than general relations of FE. When IPE is used for such general relations, formulas must be expressed in CNF or DNF form, which can result in a superpolynomial blowup in size for arbitrary formulas.

6.2. Biometric Authentication. In [24, 25], it is shown that secret key IPE can be used for biometric authentication. Consider that a company wants to limit access to certain area in a building. It can employ a biometric-based authentication system, for example, fingerprint readers or iris scanners. The biometric scanner is connected to an external authentication server. The server holds the list of employee biometric signatures and their authorization policies. However, as password-based authentication system employs salted password hashing to protect each user's password, the server should store each employee's biometric information in the form of ciphertext. Due to the noisy nature of biometrics, authentication should be successful when the provided biometric is near to a user's stored credential. It is achieved by computing the Hamming distance between them, which is the number of bits differing from each other. IPE can be employed to compute the Hamming distance between an encrypted n-bit vector $\overrightarrow{c} = (c_1, \ldots, c_n) \in \{0, 1\}^n$ and another binary vector

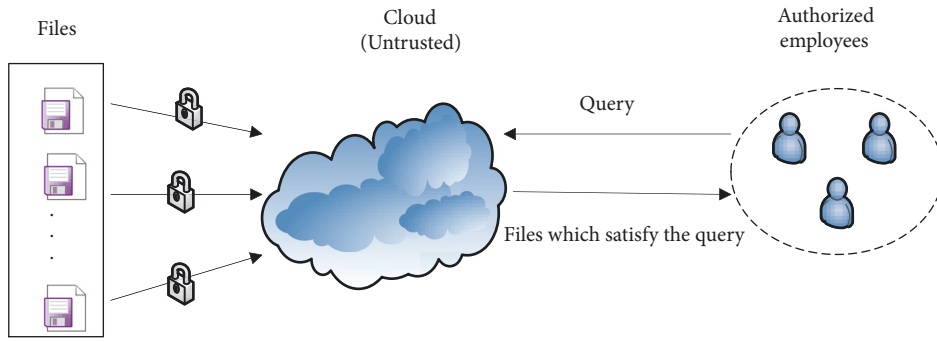

Figure 4: Search on encrypted data.

$\overrightarrow{d} = (d_1, \ldots, d_n) \in \{0,1\}^n$ embedded in the secret key. We can encode $\overrightarrow{c}$ and $\overrightarrow{d}$ as $\overrightarrow{C} = (C_1, \ldots, C_{2n}) \in \{-1,1\}^{2n}$ and $\overrightarrow{D} = (D_1, \ldots, D_{2n}) \in \{-1,1\}^{2n}$, respectively, where $C_{2i} = D_{2i} = 1$ and $C_{2i+1} = -(-1)^{c_i}$ and $D_{2i+1} = (-1)^{d_i}$ for each $i \in \{1, \ldots, n\}$. Then $\langle \overrightarrow{C}, \overrightarrow{D} \rangle = \sum_{i=1}^{n} (1 - (-1)^{c_i + d_i})$ is exactly twice the Hamming distance between $\overrightarrow{c}$ and $\overrightarrow{d}$.

Suppose an employee attempts to authenticate using a fingerprint scanner, which stores the master secret key for a secret key IPE scheme. The scanner may accomplish authentication by the secret key IPE scheme, to read the fingerprint of the employee, compute the ciphertext of the fingerprint with the master secret key, and send it to the server. The server has stored the ciphertext of each user's fingerprint under the secret key in advance. The server can compute the Hamming distance between the encrypted fingerprint and the stored fingerprint for the employee. If the result is small, authentication will succeed.

7. Conclusion

In this work, we presented an efficient simulation-based inner product encryption scheme with a polynomial-size range in the standard model. Our scheme uses asymmetric bilinear pairing groups of prime order under the XDLIN assumption. Our work raises several interesting open problems on constructions of inner product encryption schemes. One open problem is to explore the simulation-based security notion in IPE with full-hiding security, considering confidentiality for functional keys and encrypted data in a completely symmetric manner, in the secret key setting [17]. Another open problem is to study how to achieve many-AD-SIM security for a multi-input inner product functional encryption scheme with a polynomial-size range [37].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

A preliminary version of this paper was presented at the 19th International Conference on Information and Communications Security.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been partly supported by National NSF of China under Grant nos. 61772266, 61572248, and 61431008.

References

[1] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: definitions and challenges," in *Theory of Cryptography*, vol. 6597 of *Lecture Notes in Computer Science*, pp. 253–273, Springer, 2011.

[2] D. Boneh, A. Sahai, and B. Waters, "Functional encryption: A new vision for public-key cryptography," *Communications of the ACM*, vol. 55, no. 11, pp. 56–64, 2012.

[3] A. O'Neill, "Definitional issues in functional encryption," https://eprint.iacr.org/2010/556.pdf.

[4] S. Agrawal, S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption: new perspectives and lower bounds," in *Advances in cryptology-CRYPTO*, vol. 8043 of *Lecture Notes in Computer Science*, pp. 500–518, Springer, Heidelberg, Germany, 2013.

[5] Z. Brakerski and G. Segev, "Function-private functional encryption in the private-key setting," *Journal of Cryptology*, vol. 31, no. 1, pp. 202–225, 2018.

[6] A. De Caro and V. Iovino, "On the power of rewinding simulators in functional encryption," *Designs, Codes and Cryptography*, vol. 84, no. 3, pp. 373–399, 2017.

[7] A. De Caro, V. Iovino, A. Jain, A. O'Neill, O. Paneth, and G. Persiano, "On the achievability of simulation-based security for functional encryption," in *Advances in cryptology-CRYPTO*, vol. 8043 of *Lecture Notes in Computer Science*, pp. 519–535, Springer, Heidelberg, Germany, 2013.

[8] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters, "Candidate indistinguishability obfuscation and functional encryption for all circuits," in *Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 40–49, Berkeley, Calif, USA, October 2013.

[9] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich, "Reusable garbled circuits and succinct functional encryption," in *Proceedings of the 45th annual ACM symposium on Theory of computing*, pp. 555–564, Palo Alto, Calif, USA.

[10] S. Gorbunov, V. Vaikuntanathan, and H. Wee, "Functional encryption with bounded collusions via multi-party computation," in *Advances in cryptology-CRYPTO*, vol. 7417 of *Lecture Notes in Computer Science*, pp. 162–179, Springer, Heidelberg, Germany, 2012.

[11] D. I. Sharma and D. C. Jinwala, "Multiuser searchable encryption with token freshness verification," *Security and Communication Networks*, vol. 2017, Article ID 6435138, 16 pages, 2017.

[12] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Theory of cryptography*, vol. 5444 of *Lecture Notes in Computer Science*, pp. 457–473, Springer, Berlin, Germany, 2009.

[13] S. Agrawal, S. Agrawal, S. Badrinarayanan, A. Kumarasubramanian, M. Prabhakaran, and A. Sahai, "On the practical security of inner product functional encryption," in *Public-key cryptography-PKC*, vol. 9020 of *Lecture Notes in Computer Science*, pp. 777–798, Springer, Heidelberg, Germany, 2015.

[14] D. Boneh, A. Raghunathan, and G. Segev, "Function-private subspace-membership encryption and its applications," in *Advances in cryptology-ASIACRYPT*, vol. 8269 of *Lecture Notes in Computer Science*, pp. 255–275, Springer, Heidelberg, Germany, 2013.

[15] D. Boneh, A. Raghunathan, and G. Segev, "Function-private identity-based encryption: hiding the function in functional encryption," in *Advances in Cryptology – CRYPTO 2013*, vol. 8043 of *Lecture Notes in Computer Science*, pp. 461–478, 2013.

[16] A. Bishop, A. Jain, and L. Kowalczyk, "Function-hiding inner product encryption," in *Advances in cryptology-ASIACRYPT*, vol. 9452 of *Lecture Notes in Computer Science*, pp. 470–491, Springer, Heidelberg, Germany, 2015.

[17] P. Datta, R. Dutta, and S. Mukhopadhyay, "Strongly full-hiding inner product encryption," *Theoretical Computer Science*, vol. 667, pp. 16–50, 2017.

[18] J. Tomida, M. Abe, and T. Okamoto, "Efficient Functional Encryption for Inner-Product Values with Full-Hiding Security," in *Information Security Journal*, vol. 9866 of *Lecture Notes in Computer Science*, pp. 408–425, Springer International Publishing, 2016.

[19] Q. Zhao, Q. Zeng, X. Liu, and H. Xu, "Simulation-based security of function-hiding inner product encryption," *Science China Information Sciences*, vol. 61, no. 4, Article ID 048102, 3 pages, 2018.

[20] A. Sahai and H. A. Seyalioglu, "Worry-free encryption: functional encryption with public keys," in *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS'10*, pp. 463–472, Chicago, Ill, USA, October 2010.

[21] S. Garg, C. Gentry, and S. Halevi, "Candidate multilinear maps from ideal lattices," in *Advances in Cryptology—EUROCRYPT*, vol. 7881 of *Lecture Notes in Computer Science*, pp. 1–17, Springer, Berlin, Germany, 2013.

[22] B. Waters, "A punctured programming approach to adaptively secure functional encryption," in *Advances in cryptology-CRYPTO*, vol. 9216 of *Lecture Notes in Computer Science*, pp. 678–697, Springer, Heidelberg, Germany, 2015.

[23] M. Abdalla, F. Bourse, A. De Caro, and D. Pointcheval, "Simple functional encryption schemes for inner products," in *Public-key cryptography-PKC 2015*, vol. 9020 of *Lecture Notes in Computer Science*, pp. 733–751, Springer, Heidelberg, Germany, 2015.

[24] S. Agrawal, B. t. Libert, and D. Stehlé, "Fully secure functional encryption for inner products, from standard assumptions," in *Advances in cryptology-CRYPTO*, vol. 9816 of *Lecture Notes in Computer Science*, pp. 333–362, Springer, Berlin, Germany, 2016.

[25] S. Kim, K. Lewi, A. Mandal, H. Montgomery, A. Roy, and D. J. Wu, "Function-hiding inner product encryption is practical," https://eprint.iacr.org/2016/440.pdf.

[26] Q. Zhao, Q. Zeng, and X. Liu, "Efficient Inner Product Encryption with Simulation-Based Security," in *Information and Communications Security*, vol. 10631 of *Lecture Notes in Computer Science*, pp. 162–171, Springer International Publishing, 2018.

[27] J. Katz, A. Sahai, and B. Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Advances in Cryptology—EUROCRYPT*, vol. 4965 of *Lecture Notes in Computer Science*, pp. 146–162, Springer, Berlin, Germany, 2008.

[28] A. Lewko, T. Okamoto, A. Sahai, and B. Waters, "Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology: EUROCRYPT*, vol. 6110 of *Lecture Notes in Computer Science*, pp. 62–91, Springer, Berlin, Germany, 2010.

[29] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," in *Advances in Cryptology—CRYPTO*, T. Rabin, Ed., vol. 6223 of *Lecture Notes in Computer Science*, pp. 191–208, Springer, Berlin, Germany, 2010.

[30] T. Okamoto and K. Takashima, "Fully secure unbounded inner-product and attribute-based encryption," in *Advances in Cryptology—ASIACRYPT*, vol. 7658 of *Lecture Notes in Computer Science*, pp. 349–366, Springer, Berlin, Heidelberg, Germany, 2012.

[31] T. Okamoto and K. Takashima, "Homomorphic encryption and signatures from vector decomposition," in *Pairing-based cryptography-Pairing*, vol. 5209 of *Lecture Notes in Computer Science*, pp. 57–74, Springer, Berlin, Germany, 2008.

[32] T. Okamoto and K. Takashima, "Hierarchical predicate encryption for inner-products," in *Advances in Cryptology—ASIACRYPT*, vol. 5912 of *Lecture Notes in Computer Science*, pp. 214–231, Springer, Berlin, Heidelberg, Germany, 2009.

[33] M. Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki, and M. Ohkubo, "Constant-size structure-preserving signatures: generic constructions and simple assumptions," in *Advances in cryptology-ASIACRYPT 2012*, vol. 7658 of *Lecture Notes in Computer Science*, pp. 4–24, Springer, Heidelberg, Germany, 2012.

[34] J. A. Akinyele, C. Garman, I. Miers et al., "Charm: a framework for rapidly prototyping cryptosystems," *Journal of Cryptographic Engineering*, vol. 3, no. 2, pp. 111–128, 2013.

[35] "FLINT: Fast Library for Number Theory 2.5," http://flintlib.org.

[36] D. Shanks, "Class number, a theory of factorization, and genera," in *Proceedings of the Symposia in Pure Mathematics*, vol. 20, pp. 415–440, Providence, RI, USA, 1971.

[37] M. Abdalla, R. Gay, M. Raykova, and H. Wee, "Multi-input inner-product functional encryption from pairings," in *Advances in cryptology-EUROCRYPT*, vol. 10210 of *Lecture Notes in Computer Science*, pp. 601–626, Springer, 2017.