SCIS 2016 2016 Symposium on
Cryptography and Information Security
Kumamoto, Japan, Jan. 19 - 22, 2016
The Institute of Electronics,
Information and Communication Engineers

Adaptively Secure Functional Encryption for Inner-Product Values

Junichi Tomida [*] Masayuki Abe [†] Tatsuaki Okamoto [†]

Abstract: Functional encryption for inner-product values was first introduced by Abdalla et al. (PKC 2015), where decrypting a chipertext for a vector $\vec{x}$ with a secret key for a vector $\vec{y}$ reveals only the inner-product value of these vectors $\vec{x} \cdot \vec{y}$. We call it inner-product value encryption (IPVE) in contrast to inner-product predicate encryption (IPPE), which is usually called inner-product encryption (IPE). To the best of our knowledge, while there already exist a selectively secure IPVE scheme and an adaptively secure one in the secret-key setting, there is no adaptively secure one in the public-key setting. We present the first IPVE scheme that is adaptively secure in the public-key setting under the decisional linear (DLIN) assumption.

Keywords: Functional Encryption, Inner Product, Adaptive Security

1 Introduction

Functional encryption (FE) is a very useful tool for non-interactive computation on encrypted data. In a FE scheme, the owner of the master secret key msk can create a secret key sk_f for function f, and it enables users to compute the value of $f(x)$ by decrypting a ciphertext for x without revealing anything else about x. As cloud services are increasing rapidly, users' demand for computation on encrypted data is also increasing because cloud servers are by no means reliable. FE is one of the solution for this problem, providing a paradigm where users can compute a function f on encrypted data using a secret key sk_f without revealing anything else about the encrypted data to the cloud server.

Predicate encryption (PE) [5] is a subclass of FE where with a secret key sk_f for a predicate f and a ciphertext ct_I for an attribute I, a decryptor can obtain the messeage only if some relation between f and I is saticefied. We can see that PE is included in FE if we consider that $g(f,(I,m)) = m$ when $R(f,I) = 1$ and $g(f,(I,m)) = \perp$ when $R(f,I) = 0$. Identity-based encryption (IBE) [2], attribute-based encryption (ABE) [7] and inner-product encryption (IPE) [5, 6, 9, 10] are subclasses of PE. In an IPE scheme, there are a secret key $\mathsf{sk}_{\vec{v}}$ for a vector $\vec{v}$ and a ciphertext $\mathsf{ct}_{\vec{x}}$ for a vector $\vec{x}$, and it can be decrypted only if these two vectors are orthogonal (namely $\vec{v} \cdot \vec{x} = 0$).

Our scheme is different from IPE because it enables users to compute the inner-product itself of $\vec{v}$ and $\vec{x}$ (the value of $\vec{v} \cdot \vec{x}$ in $\mathbb{Z}_q$). Therefore, we call it inner-product value encryption (IPVE) in contrast to IPE or inner-product predicate encryption (IPPE). As it is written in [1], evaluation of inner-product is a useful tool for statics because it can provide the *wighted mean.* For example, it is applicable to the evaluation of grade point average.

1.1 Related Works

Abdalla et al. introduced the first IPVE scheme [1], which is constructed from selective-IND-CPA secure public-key encryption (PKE). Their scheme is selective secure and it is constructed generally from a PKE scheme that has some structural and homomorphic properties. They present instantiation from the ElGamal encryption scheme [4] and the PKE scheme from LWE introduced by Regev [11].

Bishop et al. proposed an IPVE scheme that has function privacy [3]. In a public-key based IPVE scheme, anyone can encrypt arbitrary message vectors and decrypt them by their secret key. Therefore, they can reveal the function vector embedded in the secret key with ciphertexts for the components of the canonical basis, and consequently it is impossible to ensure the function privacy. To protect privacy of functions as well as messages, they constructed a private-key based IPVE scheme using dual pairing vector spaces (DPVS) introduced by Okamoto and Takashima [8]. In their scheme, the value of inner-products assumed to be a logarithmic size because the decryptor has to compute the discrete logarithm of that. This assumption is reasonable for statistical applications, where the average or count of some bounded quantity over a logarithmically-sized database will naturally be in a logarithmic size.

1.2 Our Contribution

We construct the first adaptively secure IPVE scheme for public-key setting. Our techniques are inspired by the scheme of Bishop et al. [3] to use DPVS. The definition of the adaptive security model is different from that of private-key setting security model. Consequently, the security of our scheme is proven under deci-

[*] Graduate School of Informatics, Kyoto University
[†] NTT Secure Platform Labolatories, NTT Corporation

sional linear (DLIN) assumption whereas they use symmetric external Diffie-Hellman (SXDH) assumption for the security proof. We also assume that the value of inner-products is a logarithmic size for the same reason as their scheme.

2 Preliminaries

In this section, we present the notation and the definitions that are used in this paper.

2.1 Notation

For a set S, $x \overset{\mathsf{U}}{\leftarrow} S$ denotes that x is uniformly chosen from S. For a probability distribution X, $x \overset{\mathsf{R}}{\leftarrow} X$ denotes that x is chosen from X according to its distribution. For a prime q, $\mathbb{Z}_q$ denotes a set of integers $\{0, \cdots, q-1\}$, and $\mathbb{Z}_q^\times$ denotes a set of integers $\{1, \cdots, q-1\}$. $\vec{0}$ denotes a zero vector. For a n dimensional vector $\vec{x}$, $x_i (1 \leq i \leq n)$ denotes the i-th coordinate of $\vec{x}$. For vectors $\vec{x}$ and $\vec{y}$, $\vec{x}\cdot\vec{y}$ denotes inner-product of $\vec{x}$ and $\vec{y}$. A random linear transformation W on $\mathbb{V}$ denotes a transformation, for $\vec{x} \in \mathbb{Z}_q^n$ and $R \overset{\mathsf{U}}{\leftarrow} GL(n, \mathbb{Z}_q)$, from $(\vec{x})_\mathbb{A}$ to $(\vec{x}R)_\mathbb{A}$, i.e., $W(\vec{x})_\mathbb{A} := (\vec{x}R)_\mathbb{A}$. Similarly, $(W^{-1})^{\mathrm{T}}(\vec{x})_\mathbb{A} := (\vec{x}(R^{-1})^{\mathrm{T}})_\mathbb{A}$. For a security game and an adversary $\mathcal{A}$ for the game, $\mathsf{Exp}_\mathcal{A}^{\mathsf{Game}}(\lambda) \to b$ denotes that $\mathcal{A}$ outputs b in the game. A function $f : \mathbb{N} \to \mathbb{R}$ is *negligible* in λ, if $\forall c > 0, \exists n \in \mathbb{N}, \forall \lambda > n, f(\lambda) < \lambda^{-c}$.

2.2 Dual Pairing Vector Spaces and DLIN assumption

We will construct our IPVE scheme based on dual pairing vector spaces (DPVS) introduced in [8]. There are two types of DPVS, one is using symmetric bilinear pairing groups and the other is asymmetric bilinear pairing groups [9]. In this paper, we use the symmetric version of DPVS but we can also use the asymmetric version similarly.

Definition 1 (Symmetric bilinear pairing groups)**.** *Symmetric bilinear pairing groups are defined by the tuple $(q, \mathbb{G}, \mathbb{G}_T, G, e)$, where q is a prime, $\mathbb{G}$ and $\mathbb{G}_T$ are cyclic groups of order q, $G \neq 0 \in \mathbb{G}$ is an element of $\mathbb{G}$ and $e : \mathbb{G}\times\mathbb{G} \to \mathbb{G}_T$ is a map that has the following properties:*

1. $\forall a, b \in \mathbb{Z}_q, e(aG, bG) = e(G, G)^{ab}$
2. $e(G, G) \neq 1$.

For symmetric bilinear pairing groups, we define that $\mathsf{param}_\mathbb{G} := (q, \mathbb{G}, \mathbb{G}_T, G, e)$. We denote making an instance of symmetric bilinear pairing groups by $\mathsf{param}_\mathbb{G} \overset{\mathsf{R}}{\leftarrow} \mathsf{Gen}_{\mathsf{sbpg}}$.

Definition 2 (DPVS: Dual pairing vector spaces)**.** *DPVS are defined by the tuple $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, \tilde{e})$, which is directly constructed from $\mathsf{param}_\mathbb{G} \overset{\mathsf{R}}{\leftarrow} \mathsf{Gen}_{\mathsf{sbpg}}$. $\mathbb{V} := \mathbb{G}^n$ is an n dimensional vector space, $\mathbb{A} := (\boldsymbol{a}_1, \cdots, \boldsymbol{a}_n)$ is a canonical basis, where $\boldsymbol{a}_i := (\overbrace{0, \cdots, 0}^{i-1}, G, \overbrace{0, \cdots, 0}^{n-i})$, and $\tilde{e} : \mathbb{V}\times\mathbb{V} \to \mathbb{G}_T$ is pairing defined below. A prime q and a group $\mathbb{G}_T$ are the same entities as the instance of symmetric bilinear pairing groups. The pairing is defined by $\tilde{e}(\boldsymbol{x}, \boldsymbol{y}) := \prod_{i=1}^n e(G_i, H_i) \in \mathbb{G}_T$, where $\boldsymbol{x} := (G_1, \cdots, G_n), \boldsymbol{y} := (H_1, \cdots, H_n) \in \mathbb{V}$.*

Here we consider random dual orthonormal bases:

$$
\begin{aligned}
&\psi := \mathbb{Z}_q^\times,\\
&X := (\chi_{i,j}) \overset{\mathsf{U}}{\leftarrow} GL(n, \mathbb{Z}_q), \;\; (\vartheta_{i,j}) := \psi(X^{\mathrm{T}})^{-1}\\
&\boldsymbol{b}_i := \sum_{j=1}^n \chi_{i,j}\boldsymbol{a}_j \;\text{ for } i = 1, \cdots, n,\\
&\boldsymbol{b}_i^* := \sum_{j=1}^n \vartheta_{i,j}\boldsymbol{a}_j \;\text{ for } i = 1, \cdots, n,\\
&\mathbb{B} := (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n), \;\; \mathbb{B}^* := (\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*),\\
&g_T := e(G, G)^\psi.
\end{aligned}
$$

We denote choosing random dual orthonormal sets this way by $(\mathbb{B}, \mathbb{B}^*) \overset{\mathsf{R}}{\leftarrow} \mathsf{Dual}(\mathbb{Z}_q^n, \psi)$, and $(\mathsf{param}_\mathbb{G}, g_T)$ by $\mathsf{param}_\mathbb{V}$. For a vector $\vec{x} := (x_1, \cdots, x_n) \in \mathbb{Z}_q^n$ and a basis $\mathbb{B} := (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n)$ on $\mathbb{V}$, we denote $\sum_{i=1}^n x_i\boldsymbol{b}_i$ by $(\vec{x})_\mathbb{B}$. Then it can be seen that

$$
\begin{aligned}
\tilde{e}((\vec{x})_\mathbb{A}, (\vec{y})_\mathbb{A}) := \prod_{i=1}^n e(x_iG, y_iG) =& e(G, G)^{\sum_{i=1}^n x_iy_i}\\
=& e(G, G)^{\vec{x}\cdot\vec{y}}.
\end{aligned}
$$

Consequently,

$$
\begin{aligned}
\tilde{e}((\vec{x})_\mathbb{B}, (\vec{y})_{\mathbb{B}^*}) = \tilde{e}((\vec{x}X)_\mathbb{A}, (\vec{y}(X^{\mathrm{T}})^{-1})_\mathbb{A}) =& e(G, G)^{\psi\vec{x}\cdot\vec{y}}\\
=& g_T^{\vec{x}\cdot\vec{y}}.
\end{aligned}
$$

Definition 3 (DLIN: Decisional linear assumption)**.** *The DLIN problem is to guess $b \in \{0, 1\}$, given $\mathsf{param}_\mathbb{G} \overset{\mathsf{R}}{\leftarrow} \mathsf{Gen}_{\mathsf{sbpg}}$ and the tupple $(\xi G, \kappa G, \delta\xi G, \sigma\kappa G, Y_b)$, where $\xi, \kappa, \delta, \sigma \overset{\mathsf{U}}{\leftarrow} \mathbb{Z}_q, Y_0 = (\delta + \sigma)G, Y_1 \overset{\mathsf{U}}{\leftarrow} \mathbb{G}$ and $b \overset{\mathsf{U}}{\leftarrow} \{0, 1\}$. The DLIN assumption is that, for any probabilistic polynomial-time (PPT) adversary $\mathcal{B}$, the advantage to DLIN problem defined below is negligible in λ:*

$$\mathsf{Adv}_\mathcal{B}^{\mathsf{DLIN}}(\lambda) := \left|\Pr[\mathcal{B}(1^\lambda, P_0) \to 1] - \Pr[\mathcal{B}(1^\lambda, P_1) \to 1]\right|$$

where $P_b := (\mathsf{param}_\mathbb{G}, \xi G, \kappa G, \delta\xi G, \sigma\kappa G, Y_b)$ is an instance of the DLIN problem defined above.

2.3 Definitions of IPVE and its security

Definition 4 (IPVE: Inner-product value encryption)**.** *An IPVE scheme scheme consists of four PPT algorithms* Setup*,* KeyGen*,* Enc *and* Dec*:*

- $\mathsf{Steup}(1^\lambda, n)$: *A setup algorithm that takes as input a security parameter 1^λ and a vector length parameter n (a positive integer that is polynomial in λ), and outputs a master public key* mpk *and a master secret key* msk*.*

- $\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \vec{y})$: *A key generation algorithm that takes as input a master public key* mpk*, a master secret key* msk *and a key vector* $\vec{y} \in \mathbb{Z}_q^n \backslash \{\vec{0}\}$*, and outputs a corresponding secret key* $\mathsf{sk}_{\vec{y}}$*.*
- $\mathsf{Enc}(\mathsf{mpk}, \vec{x})$: *An encryption algorithm that takes as input a master public key* mpk *and a message vector* $\vec{x} \in \mathbb{Z}_q^n$*, and outputs a ciphertext* $\mathsf{ct}_{\vec{x}}$*.*
- $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\vec{y}}, \mathsf{ct}_{\vec{x}})$: *A decryption algorithm that akes as input a master public key* mpk*, a secret key* $\mathsf{sk}_{\vec{y}}$ *and a ciphertext* $\mathsf{ct}_{\vec{x}}$*, and outputs ether a value* $m \in \mathbb{Z}_q$ *or a symbol* $\perp$*.*

We assume the following properties for correctness: for all $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\mathsf{R}} \mathsf{Steup}(1^\lambda, n)$, all $\vec{x} \xleftarrow{\mathsf{U}} \mathbb{Z}_q^n$, all $\vec{y} \xleftarrow{\mathsf{U}} \mathbb{Z}_q^n \backslash \{\vec{0}\}$, all $\mathsf{sk}_{\vec{y}} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \vec{y})$ and all $\mathsf{ct}_{\vec{x}} \xleftarrow{\mathsf{R}} \mathsf{Enc}(\mathsf{mpk}, \vec{x})$, if the value of $\vec{x} \cdot \vec{y}$ is logarithmic size, $\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\vec{y}}, \mathsf{ct}_{\vec{x}})$ must output $m = \vec{x} \cdot \vec{y}$. Otherwise, it outputs a symbol $\perp$.

Definition 5 (Adaptive security)**.** *An IPVE scheme is adaptvely secure if, for any PPT adversaries* $\mathcal{A}$*, the advantage in the following game is negligible in* λ*:*

1. *The challenger runs* $\mathsf{Steup}(1^\lambda, n)$ *to generate* mpk *and* msk*, and gives* mpk *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *may adaptively query the challenger with secret keys for any vectors* $\vec{y} \in \mathbb{Z}_q^n \backslash \{\vec{0}\}$ *polynomial times. The challenger replies to* $\mathcal{A}$ *with corresponding secret keys* $\mathsf{sk}_{\vec{y}} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \vec{y})$*.*
3. $\mathcal{A}$ *outputs challenge vectors* $\vec{x}_0, \vec{x}_1 \in \mathbb{Z}_q^n$ *such that* $\vec{x}_0 \cdot \vec{y} = \vec{x}_1 \cdot \vec{y}$ *for all queried* $\vec{y}$ *and submits them to the challenger. The challenger chooses a random bit* b *and gives a ciphertext* $\mathsf{ct}_{\vec{x}_b} \xleftarrow{\mathsf{R}} \mathsf{Enc}(\mathsf{mpk}, \vec{x}_b)$ *to* $\mathcal{A}$*.*
4. $\mathcal{A}$ *may adaptively query the challenger with secret keys for any vectors* $\vec{y} \in \mathbb{Z}_q^n \backslash \{\vec{0}\}$ *such that* $\vec{x}_0 \cdot \vec{y} = \vec{x}_1 \cdot \vec{y}$ *polynomial times. The challenger replies to* $\mathcal{A}$ *with corresponding secret keys* $\mathsf{sk}_{\vec{y}} \xleftarrow{\mathsf{R}} \mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \vec{y})$*.*
5. $\mathcal{A}$ *outputs a bit* b' *as the conjecture of* b*.*

The advantage of $\mathcal{A}$ *in this game is defined as*

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AS,IPVE}}(\lambda) := \left| \begin{array}{l} \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game0}}(\lambda) \to 1] \\ -\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game1}}(\lambda) \to 1] \end{array} \right|$$

where this game is defined as $\mathsf{game}\ 0$ *if* $b = 0$ *and as* $\mathsf{game}\ 1$ *if* $b = 1$*.*

To state more precisely, this security definition indicates *adaptive indistinguishablity against chosen plainext attacks.*

3 Construction

We will present our adaptive secure IPVE scheme in this section. We use $n + 3$ dimensional DPVS for encoding n dimensional key vectors $\vec{y}$ and message vectors $\vec{x}$. By pairing of a secret key $\mathsf{sk}_{\vec{y}}$ and a ciphertext $\mathsf{ct}_{\vec{x}}$, the decryptor can obtain the inner-product value $\vec{x} \cdot \vec{y}$ as a exponent. Then the decryptor computes discrete logarithm for it and get the inner-product value.

$\mathsf{Steup}(1^\lambda, n)$

The setup algorithm makes $\mathsf{param}_{\mathbb{G}} \xleftarrow{\mathsf{R}} \mathsf{Gen}_{\mathsf{sbpg}}$, and random dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{Z}_q^{n+3}, \psi)$. It defines

$$\begin{aligned} &\widehat{\mathbb{B}} := (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{n+2}), \\ &\widehat{\mathbb{B}}^* := (\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*), \\ &\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\mathbb{V}}, \widehat{\mathbb{B}}), \quad \mathsf{msk} := \widehat{\mathbb{B}}^*. \end{aligned}$$

Then it outputs $(\mathsf{mpk}, \mathsf{msk})$.

$\mathsf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \vec{y})$

The key generation algorithm computes a secret key $\mathsf{sk}_{\vec{y}}$ as

$$\mathsf{sk}_{\vec{y}} := (\vec{y}, 0, 0, 0)_{\mathbb{B}^*},$$

and outputs it.

$\mathsf{Enc}(\mathsf{mpk}, \vec{x})$

The encryption algorithm computes a ciphertext $\mathsf{ct}_{\vec{x}}$ as

$$\begin{aligned} &\alpha, \phi \xleftarrow{\mathsf{U}} \mathbb{Z}_q, \\ &\mathsf{ct}_{\vec{x}} := (\vec{x}, \alpha, \phi, 0)_{\mathbb{B}}, \end{aligned}$$

and outputs it.

$\mathsf{Dec}(\mathsf{mpk}, \mathsf{sk}_{\vec{y}}, \mathsf{ct}_{\vec{x}})$

The decryption algorithm computes the pairing of a secret key and a ciphertext as

$$d := \tilde{e}(\mathsf{ct}_{\vec{x}}, \mathsf{sk}_{\vec{y}}).$$

Then it computes m such that $g_T^m = d$ in the polynomial range fixed in advance. If it finds m that satisfies $g_T^m = d$, outputs m. Otherwise, outputs $\perp$.

Correctness

$$d := \tilde{e}(\mathsf{ct}_{\vec{x}}, \mathsf{sk}_{\vec{y}}) = g_T^{\vec{x} \cdot \vec{y}}$$

Therefore, m that the decryption algorithm outputs is $\vec{x} \cdot \vec{y}$.

Remark. Our scheme satisfies adaptive security in Definition 5, but we can easily see that it is malleable. Consequently, we desire to transform it into a CCA secure scheme and it is possible to do in the same way as Section 9 in [9].

4 Security proof

In this section, we prove that our IPVE scheme is adaptively secure under the DLIN assumption.

Theorem 1. *The proposed IPVE scheme is adaptively secure under the DLIN assumption. For any PPT adversary $\mathcal{A}$, there exists a PPT algorithm $\mathcal{B}$, such that for any security parameter λ,*

$$\mathsf{Adv}^{\mathsf{AS,IPVE}}_{\mathcal{A}}(\lambda) \le 2\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{B}}(\lambda) + \frac{10}{q}.$$

For the proof of Theorem 1, we consider following two problems.

Problem 1. *Problem 1 is to guess a bit b, given* $(\mathsf{param}_{\mathbb{V}}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \boldsymbol{f}_b)$*, where*

$$\begin{aligned}
&\mathsf{param}_{\mathbb{G}} \xleftarrow{\mathsf{R}} \mathsf{Gen}_{\mathsf{sbpg}}, \quad (\mathbb{B}, \mathbb{B}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{Z}_q^{n+3}, \psi),\\
&\widehat{\mathbb{B}} := (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{n+2}), \quad \widehat{\mathbb{B}}^* := (\boldsymbol{b}_1^*, \cdots, \boldsymbol{b}_n^*),\\
&\alpha, \phi \xleftarrow{\mathsf{U}} \mathbb{Z}_q, \quad \tau \xleftarrow{\mathsf{U}} \mathbb{Z}_q^{\times},\\
&\boldsymbol{f}_0 := (0^n, \alpha, \phi, 0)_{\mathbb{B}}, \quad \boldsymbol{f}_1 := (0^n, \alpha, \phi, \tau)_{\mathbb{B}}.
\end{aligned}$$

For a PPT algorithm $\mathcal{C}$, the advantage of $\mathcal{C}$ for Probrem 1 is defined as

$$\mathsf{Adv}^{\mathsf{P1}}_{\mathcal{C}}(\lambda) := \left|\mathsf{Pr}[\mathcal{C}(1^\lambda, P_0) \to 1] - \mathsf{Pr}[\mathcal{C}(1^\lambda, P_1) \to 1]\right|$$

where P_b is an instance of the Problem 1 defined above.

Problem 2 (Definition 18 in [9])**.** *Problem 2 is to guess a bit b, given* $(\mathsf{param}_{\mathsf{P2}}, \widehat{\mathbb{B}}, \mathbb{B}^*, \boldsymbol{y}_b^*, \kappa G, \xi G)$*, where*

$$\begin{aligned}
&\mathsf{param}_{\mathbb{G}} \xleftarrow{\mathsf{R}} \mathsf{Gen}_{\mathsf{sbpg}},\\
&X := (\chi_{i,j}) \xleftarrow{\mathsf{U}} GL(3, \mathbb{Z}_q), \quad (\vartheta_{i,j}) := (X^{\mathrm{T}})^{-1},\\
&\kappa, \xi \xleftarrow{\mathsf{U}} \mathbb{Z}_q^{\times},\\
&\boldsymbol{b}_i := \kappa \sum_{j=1}^{3} \chi_{i,j} \boldsymbol{a}_j \quad \text{for } i = 1, 2, 3,\\
&\boldsymbol{b}_i^* := \xi \sum_{j=1}^{3} \vartheta_{i,j} \boldsymbol{a}_j \quad \text{for } i = 1, 2, 3,\\
&\widehat{\mathbb{B}} := (\boldsymbol{b}_1, \boldsymbol{b}_3), \quad \mathbb{B}^* := (\boldsymbol{b}_1^*, \boldsymbol{b}_2^*, \boldsymbol{b}_3^*),\\
&g_T := e(G, G)^{\kappa\xi}, \mathsf{param}_{\mathsf{P2}} := (\mathsf{param}_{\mathbb{G}}, g_T),\\
&\delta, \sigma \xleftarrow{\mathsf{U}} \mathbb{Z}_q, \quad \rho \xleftarrow{\mathsf{U}} \mathbb{Z}_q^{\times},\\
&\boldsymbol{y}_0^* := (\delta, 0, \sigma)_{\mathbb{B}^*}, \quad \boldsymbol{y}_1^* := (\delta, \rho, \sigma)_{\mathbb{B}^*}.
\end{aligned}$$

For a PPT algorithm $\mathcal{D}$, the advantage of $\mathcal{D}$ for Problem 2 is defined in the same way as Problem 1.

Lemma 1. *For any PPT adversary $\mathcal{D}$ for Problem2, there exists a PPT algorithm $\mathcal{B}$ for the DLIN problem, such that for any security parameter λ,* $\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{D}}(\lambda) \le \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{B}}(\lambda) + 5/q$.

Proof. Demonstrated at Lemma 15 in [9]. □

Lemma 2. *For any PPT adversary $\mathcal{C}$ for Problem1, there exists a PPT algorithm $\mathcal{D}$ for Problem 2, such that for any security parameter λ,* $\mathsf{Adv}^{\mathsf{P1}}_{\mathcal{C}}(\lambda) \le \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{D}}(\lambda)$.

Proof. $\mathcal{D}$ is give an instance of Problem 2 $(\mathsf{param}_{\mathsf{P2}}, \widehat{\mathbb{B}}, \mathbb{B}^*, \boldsymbol{y}_b^*, \kappa G, \xi G)$. $\mathcal{D}$ generates a random linear transformation W on $\mathbb{G}^{n+3}$, and sets

$$\begin{aligned}
&\mathsf{param}_{\mathbb{V}} := \mathsf{param}_{\mathsf{P2}},\\
&\boldsymbol{d}_i := W(0^{i+2}, \xi G, 0^{n-i}) \quad \text{for } i = 1, \cdots, n,\\
&\boldsymbol{d}_{n+1} := W(\boldsymbol{b}_1^*, 0^n), \quad \boldsymbol{d}_{n+2} := W(\boldsymbol{b}_3^*, 0^n),\\
&\boldsymbol{d}_{n+3} := W(\boldsymbol{b}_2^*, 0^n),\\
&\boldsymbol{d}_i^* := (W^{-1})^{\mathrm{T}}(0^{i+2}, \kappa G, 0^{n-i}) \quad \text{for } i = 1, \cdots, n,\\
&\boldsymbol{d}_{n+1}^* := (W^{-1})^{\mathrm{T}}(\boldsymbol{b}_1, 0^n), \quad \boldsymbol{d}_{n+2}^* := (W^{-1})^{\mathrm{T}}(\boldsymbol{b}_3, 0^n),\\
&\boldsymbol{d}_{n+3}^* := (W^{-1})^{\mathrm{T}}(\boldsymbol{b}_2, 0^n),\\
&\boldsymbol{g}_b := W(\boldsymbol{y}_b^*, 0^n),\\
&\mathbb{D} \ := (\boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n+3}), \quad \mathbb{D}^* := (\boldsymbol{d}_1^*, \cdots, \boldsymbol{d}_{n+3}^*).
\end{aligned}$$

We can see that $(\mathbb{D}, \mathbb{D}^*)$ are dual orthonormal bases. $\mathcal{D}$ does not have $\boldsymbol{b}_2$ but it can compute

$$\widehat{\mathbb{D}} := (\boldsymbol{d}_1, \cdots, \boldsymbol{d}_{n+2}), \quad \widehat{\mathbb{D}}^* := (\boldsymbol{d}_1^*, \cdots, \boldsymbol{d}_n^*).$$

Then $\mathcal{D}$ gives $(\mathsf{param}_{\mathbb{V}}, \widehat{\mathbb{D}}, \widehat{\mathbb{D}}^*, \boldsymbol{g}_b)$ to $\mathcal{C}$, and outputs b' if $\mathcal{C}$ outputs b'. We can see that

$$\boldsymbol{g}_0 := (0^n, \alpha', \phi', 0)_{\mathbb{D}}, \quad \boldsymbol{g}_1 := (0^n, \alpha', \phi', \tau')_{\mathbb{D}},$$

where $\alpha' := \delta$, $\phi' := \sigma$ and $\tau' := \rho$. It is the same as Problem 1. □

Proof of Theorem 1. We employ four sequent security games, Game 0, Game 0', Game 1' and Game 1 in the proof. A framed coefficients by a box indicates they were changed from the previous game.

Game 0 This game is a original one where the challenger reply to the adversary with a chiphertext for $\vec{x}_0$. Namely, the reply to the j-th key query for $\vec{y}^{(j)}$ is

$$\mathsf{sk}_{\vec{y}}^{(j)} := (\vec{y}^{(j)}, 0, 0, 0)_{\mathbb{B}^*},$$

and the challenge ciphertext for vectors $\vec{x}_0, \vec{x}_1$ is

$$\mathsf{ct}_{\vec{x}} := (\vec{x}_0, \alpha, \phi, 0)_{\mathbb{B}},$$

where $\alpha, \phi \xleftarrow{\mathsf{U}} \mathbb{Z}_q$.

Game 0' This game is the same as Game 0 except that the challenge ciphertext for vectors $\vec{x}_0, \vec{x}_1$ is

$$\mathsf{ct}_{\vec{x}} := (\vec{x}_0, \alpha, \phi, \boxed{\tau})_{\mathbb{B}},$$

where $\alpha, \phi \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and $\tau \xleftarrow{\mathsf{U}} \mathbb{Z}_q^{\times}$.

Game 1' This game is the same as Game 0' except that the challenge ciphertext for vectors $\vec{x}_0, \vec{x}_1$ is

$$\mathsf{ct}_{\vec{x}} := (\boxed{\vec{x}_1}, \alpha, \phi, \tau)_{\mathbb{B}},$$

where $\alpha, \phi \xleftarrow{\mathsf{U}} \mathbb{Z}_q$ and $\tau \xleftarrow{\mathsf{U}} \mathbb{Z}_q^{\times}$.

Game 1 This game is a original one where the challenger reply to the adversary with a chiphertext for $\vec{x}_1$. Namely, it is the same as Game 1′ except that the challenge ciphertext for vectors $\vec{x}_0, \vec{x}_1$ is

$$\mathsf{ct}_{\vec{x}} := (\vec{x}_1, \alpha, \phi, \boxed{0})_{\mathbb{B}},$$

where $\alpha, \phi \xleftarrow{\mathsf{U}} \mathbb{Z}_q$.

Lemma 3. *For any PPT distinguisher $\mathcal{E}$ between* Game 0 *and* Game 0′*, there exists a PPT algorithm $\mathcal{C}$ for Problem 1, such that for any security parameter λ,*

$$\left|\Pr[\mathsf{Exp}_{\mathcal{E}}^{\mathsf{Game0}}(\lambda) \to 1] - \Pr[\mathsf{Exp}_{\mathcal{E}}^{\mathsf{Game0}'}(\lambda) \to 1]\right| \leq \mathsf{Adv}_{\mathcal{C}}^{\mathsf{P1}}(\lambda)$$

Proof. We will demonstrate that it is possible to construct a PPT algorithm $\mathcal{C}$ for Problem 1 using any PPT distinguisher $\mathcal{E}$ between Game 0 and Game 0′ as a blackbox:

1. $\mathcal{C}$ is given a Problem 1 instance$(\mathsf{param}_{\mathbb{V}}, \widehat{\mathbb{B}}, \widehat{\mathbb{B}}^*, \boldsymbol{f}_b)$.

2. $\mathcal{C}$ gives $(1^\lambda, \mathsf{param}_{\mathbb{V}}, \widehat{\mathbb{B}})$ to $\mathcal{E}$ as mpk.

3. $\mathcal{C}$ computes $\mathsf{sk}_{\vec{y}}$ using $\widehat{\mathbb{B}}^*$ when $\mathcal{E}$ queries $\vec{y}$, and give it to $\mathcal{E}$.

4. $\mathcal{C}$ computes $\mathsf{ct}_{\vec{x}}$ when $\mathcal{E}$ submits challenge vectors $\vec{x}_0 := (x_{0,1}, \cdots, x_{0,n}), \vec{x}_1$ as

   $$\mathsf{ct}_{\vec{x}} := \boldsymbol{f}_b + \sum_{i=1}^{n} x_{0,i}\boldsymbol{b}_i,$$

   and give it to $\mathcal{E}$.

5. $\mathcal{E}$ outputs b' as a conjecture of b, and $\mathcal{C}$ outputs b' as it is.

It can be seen that if $b = 0$, $\mathcal{E}$'s view is the same as that in Game 0, and if $b = 1$, $\mathcal{E}$'s view is the same as that in Game 0′. □

Lemma 4. *For any PPT distinguisher $\mathcal{F}$ between* Game 0′ *and* Game 1′*,*

$$\Pr[\mathsf{Exp}_{\mathcal{F}}^{\mathsf{Game0}'}(\lambda) \to 1] = \Pr[\mathsf{Exp}_{\mathcal{F}}^{\mathsf{Game1}'}(\lambda) \to 1].$$

Proof. We will demonstrate that $\mathcal{F}$'s view in Game 0′ is the same as that in Game 1′. For that purpose, we define new bases $(\mathbb{F}, \mathbb{F}^*)$ on $\mathbb{G}^{n+3}$ such that

$$\begin{aligned}
\boldsymbol{f}_{n+3} &:= \boldsymbol{b}_{n+3} + \sum_{i=1}^{n} \frac{x_{0,i} - x_{1,i}}{\tau}\boldsymbol{b}_i \\
\boldsymbol{f}_i^* &:= \boldsymbol{b}_i^* - \frac{x_{0,i} - x_{1,i}}{\tau}\boldsymbol{b}_{n+3}^* \quad \text{for } i = 1, \cdots, n, \\
\mathbb{F} &:= (\boldsymbol{b}_1, \cdots, \boldsymbol{b}_{n+2}, \boldsymbol{f}_{n+3}), \\
\mathbb{F}^* &:= (\boldsymbol{f}_1^*, \cdots, \boldsymbol{f}_n^*, \boldsymbol{b}_{n+1}^*, \boldsymbol{b}_{n+2}^*, \boldsymbol{b}_{n+3}^*),
\end{aligned}$$

where $x_{b,i}$ for $b = 0, 1$ denotes the i-th coordinate of $\vec{x}_b$. We can see that $(\mathbb{F}, \mathbb{F}^*)$ are dual orthonormal bases and they are distributed the same as $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{\mathsf{R}} \mathsf{Dual}(\mathbb{Z}_q^{n+3}, \psi)$. Then,

$$\begin{aligned}
&\mathsf{ct}_{\vec{x}} = (\vec{x}_0, \alpha, \phi, \tau)_{\mathbb{B}} \\
&= \sum_{i=1}^{n} x_{0,i}\boldsymbol{b}_i + \alpha\boldsymbol{b}_{n+1} + \phi\boldsymbol{b}_{n+2} + \tau\boldsymbol{b}_{n+3} \\
&= \sum_{i=1}^{n} x_{0,i}\boldsymbol{b}_i + \alpha\boldsymbol{b}_{n+1} + \phi\boldsymbol{b}_{n+2} + \tau\left(\boldsymbol{f}_{n+3} - \sum_{i=1}^{n} \frac{x_{0,i} - x_{1,i}}{\tau}\boldsymbol{b}_i\right) \\
&= \sum_{i=1}^{n} x_{1,i}\boldsymbol{b}_i + \alpha\boldsymbol{b}_{n+1} + \phi\boldsymbol{b}_{n+2} + \tau\boldsymbol{f}_{n+3} \\
&= (\vec{x}_1, \alpha, \phi, \tau)_{\mathbb{F}}.
\end{aligned}$$

On the other hand, the key that $\mathcal{F}$ has received as the reply for the j-th query is

$$\begin{aligned}
k_1^{(j)} &= (\vec{y}^{(j)}, 0, 0, 0)_{\mathbb{B}^*} \\
&= \sum_{i=1}^{n} y_i^{(j)}\boldsymbol{b}_i^* \\
&= \sum_{i=1}^{n} y_i^{(j)}\left(\boldsymbol{f}_i^* + \frac{x_{0,i} - x_{1,i}}{\tau}\boldsymbol{b}_{n+3}^*\right) \\
&= \sum_{i=1}^{n} y_i^{(j)}\boldsymbol{f}_i^* + \frac{1}{\tau}\sum_{i=1}^{n} y_i^{(j)}(x_{0,i} - x_{1,i})\boldsymbol{b}_{n+3}^* \\
&= \sum_{i=1}^{n} y_i^{(j)}\boldsymbol{f}_i^* + \frac{\vec{y}^{(j)}(\vec{x}_0 - \vec{x}_1)}{\tau}\boldsymbol{b}_{n+3}^* \\
&= \sum_{i=1}^{n} y_i^{(j)}\boldsymbol{f}_i^* \qquad \because \vec{x}_0 \cdot \vec{y}^{(j)} = \vec{x}_1 \cdot \vec{y}^{(j)} \\
&= (\vec{y}^{(j)}, 0, 0, 0)_{\mathbb{F}^*}.
\end{aligned}$$

Therefore, $\mathcal{F}$'s view in both game is information-theoretically identical. □

Lemma 5. *For any PPT distinguisher $\mathcal{G}$ between* Game 1′ *and* Game 1*, there exists a PPT algorithm $\mathcal{C}$ for Problem 1, such that for any security parameter λ,*

$$\left|\Pr[\mathsf{Exp}_{\mathcal{G}}^{\mathsf{Game1}'}(\lambda) \to 1] - \Pr[\mathsf{Exp}_{\mathcal{G}}^{\mathsf{Game1}}(\lambda) \to 1]\right| \leq \mathsf{Adv}_{\mathcal{C}}^{\mathsf{P1}}(\lambda).$$

Proof. The proof of Lemma 5 is similar to that of Lemma 3. □

From Lemma 1 to 5,

$$\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{AS,IPVE}}(\lambda) &= \left|\begin{array}{l} \Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game0}}(\lambda) \to 1] \\ -\Pr[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{Game1}}(\lambda) \to 1] \end{array}\right| \\
&\leq \left|\Pr[\mathsf{Exp}_{\mathcal{E}}^{\mathsf{Game0}}(\lambda) \to 1] - \Pr[\mathsf{Exp}_{\mathcal{E}}^{\mathsf{Game0}'}(\lambda) \to 1]\right| \\
&+ \left|\Pr[\mathsf{Exp}_{\mathcal{F}}^{\mathsf{Game0}'}(\lambda) \to 1] - \Pr[\mathsf{Exp}_{\mathcal{F}}^{\mathsf{Game1}'}(\lambda) \to 1]\right| \\
&+ \left|\Pr[\mathsf{Exp}_{\mathcal{G}}^{\mathsf{Game1}'}(\lambda) \to 1] - \Pr[\mathsf{Exp}_{\mathcal{G}}^{\mathsf{Game1}}(\lambda) \to 1]\right| \\
&\leq 2\mathsf{Adv}_{\mathcal{C}}^{\mathsf{P1}}(\lambda) \leq 2\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DLIN}}(\lambda) + \frac{10}{q}.
\end{aligned}$$

□

5 Conclusion

In this paper, we presented the first adaptively secure IPVE scheme in the public-key setting. Our scheme has a limitation on a size of inner-product because we have to compute the discrete logarithm of that. More work for overcoming this limitation is needed to get efficient functional computation and extend its application.

References

[1] M. Abdalla, F. Bourse, A. D. Caro, and D. Pointcheval. Simple functional encryption schemes for inner products. *In PKC*, pages 733-751. Springer, 2015.

[2] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213-229. Springer, 2001.

[3] A. Bishop, A. Jain, and L. Kowalczyk. Function-hiding inner product encryption. Cryptology ePrint Archive, Report 2015/672, 2015.

[4] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions on Information Theory*, 31:469-472, 1985.

[5] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunc- tions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146-162. Springer, 2008.

[6] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62-91. Springer, 2010.

[7] R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non- monotonic access structures. *ACM Conference on Computer and Communications Security*, pages 195-203. ACM, 2007.

[8] T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, pages 214-231. Springer, 2009.

[9] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191-208. Springer, 2010.

[10] T. Okamoto and K. Takashima. Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In *EUROCRYPTO*, pages 591-608. Springer, 2012.

[11] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *37th ACM STOC*, pages 84-93. ACM Press, 2005.