# Spatial Encryption Revisited: From Delegatable Multiple Inner Product Encryption and More

Huy Quoc Le[1,2(✉)], Dung Hoang Duong[1(✉)], Willy Susilo[1(✉)] and Josef Pieprzyk[2,3(✉)]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Northfields Avenue, Wollongong NSW 2522, Australia.
qhl576@uowmail.edu.au, {hduong,wsusilo}@uow.edu.au,
[2] CSIRO Data61, Sydney, NSW, Australia,
[3] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland.
Josef.Pieprzyk@data61.csiro.au

**Abstract.** Spatial Encryption (SE), which involves encryption and decryption with affine/vector objects, was introduced by Boneh and Hamburg at Asiacrypt 2008. Since its introduction, SE has been shown as a versatile and elegant tool for implementing many other important primitives such as (Hierarchical) Identity-based Encryption ((H)IBE), Broadcast (H)IBE, Attribute-based Encryption, and Forward-secure cryptosystems.

This paper revisits SE toward a more compact construction in the lattice setting. In doing that, we introduce a novel primitive called Delegatable Multiple Inner Product Encryption (DMIPE). It is a delegatable generalization of Inner Product Encryption (IPE) but different from the Hierarchical IPE (HIPE) (Okamoto and Takashima at Asiacrypt 2009). We point out that DMIPE and SE are equivalent in the sense that there are security-preserving conversions between them. As a proof of concept, we then successfully instantiate a concrete DMIPE construction relying on the hardness of the decisional learning with errors problem. In turn, the DMIPE design implies a more compact lattice-based SE in terms of sizes compared with SEs converted from HIPE (e.g., Xagawa's HIPE at PKC 2013) using the framework by Chen et al. (Designs, Codes, and Cryptography, 2014). Furthermore, we demonstrate that one can also use SE to implement the Allow-/Deny-list encryption, which subsumes, e.g., puncturable encryption (Green and Miers at IEEE S&P 2015).

***Key words:*** spatial encryption, learning with errors, inner product encryption, delegatable multiple inner product encryption, hierarchical inner product encryption, allow-/deny-list encryption, lattice evaluation, lattice trapdoors

## 1 Introduction

Predicate encryption (PrE) introduced by Katz, Sahai and Waters [25] generalizes identity-based encryption (IBE) [30], attribute-based encryption (ABE) [21] and hidden vector encryption (HVE) [8]. Roughly speaking, in PrE, decryption keys and ciphertexts are associated with *predicates* and *attributes*, respectively. One can consider a predicate as a function and an attribute as a variable. Assume that one wants to decrypt a ciphertext $\mathsf{ct}_{\mathbf{x}}$ respective to an attribute $\mathbf{x}$, using a decryption key $\mathsf{sk}_f$ respective to a predicate $f$. Then, the decryption is successful only if $f(\mathbf{x}) = 1$ holds. Besides IBE, ABE and HVE, PrE also covers some other classes of encryption such as spatial encryption (SE) [7], for instance.

Spatial Encryption (SE) was introduced by Boneh and Hamburg in their paper [7] at Asiacrypt 2008, and then it was systematically investigated in Hamburg's thesis [23]. Particularly, an SE predicate is an affine space/vector space, say $f := \mathbf{V}$, and an SE attribute is an affine point/vector, say $\mathbf{x} := \mathbf{v}$. The condition $f(\mathbf{x}) = 1$ is equivalent to $\mathbf{v} \in \mathbf{V}$. Furthermore, in SE one can use a decryption key for a predicate, say $\mathbf{V}_1$, to delegate a decryption key for any predicate, say $\mathbf{V}_2$, which is a subspace of $\mathbf{V}_1$. In the case of SE involving affine objects, we call it *affine SE* and in the case of SE involving vector objects–call it *linear SE*. However, as we will show later, an affine SE can be transformed into a linear SE, then it is sufficient to talk about linear SEs throughout this work.

There also exists a variant of SE called doubly spatial encryption (DSE), which is expected to be more expressive than SE. In DSE, *attribute spaces* (i.e., vector spaces or affine spaces) are needed for encryption rather than vectors. Decryption is successful if the intersection of the attribute and policy spaces is not empty. Because of essential applications of SE (and DSE) to constructing other cryptographic primitives (that will be discussed further in this work; now the readers can have a look at Figure 1 for the relation of SE with some of them), SE and DSE have been the main topic of many research works [7], [23], [35], [11], [12], [13].

### 1.1 Our Motivations

Our work is inspired by a wide range of possible applications of SE and DSE, as argued in [7] and [23]. However, the main driver behind our work is an attempt to remove the shortcomings of a generic SE construction via [12]. Furthermore, as lattice-based cryptosystems are resistant to quantum adversaries, we propose a post-quantum lattice-based SE construction, which is more efficient than other lattice-based ones, such as those from [1], [34].

**Applications of SE.** SE can be used to build many other cryptographic primitives such as (H)IBE, broadcast (H)IBE, and encryption schemes with forward security (see [7], [23]). This is done by converting e.g., identities and time periods into vectors/spaces compatible with SE. For more details, the reader is referred to [7], [23]. Our further discussion is driven by the question: "*Can we use SE to implement other important cryptographic primitives*?" We have discovered some new applications of SE. It turns out that we can use SE to construct

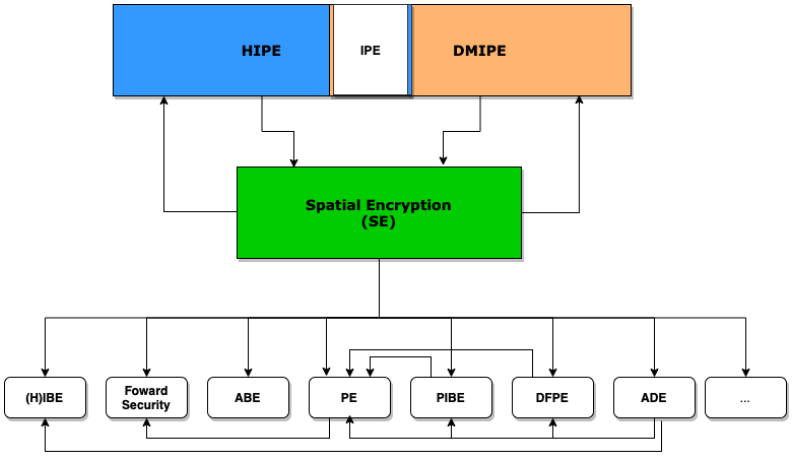


Fig. 1: The relation of SE and other primitives. Here, the implying relation A $\rightarrow$ B means that from A one can construct B.

puncturable encryption (PE) [22], [32], DFPE [16] and their generalization, the allow/deny-list encryption (ADE). Remark that ADE has also been mentioned by Derler et al. in their work [16] and can also subsume other "predecessors" such as IBE [30], HIBE [20], Fully Puncturable Encryption (FuPE) [15], Puncturable IBE (PIBE) [17]. We refer to [16, Table 1] for a summary and comparison of ADE and its predecessors. Note that, however, so far, there is neither a formal definition nor security notions for ADE.

**Lattice-based SE.** There are many instantiations of SE in the literature (see Table 1). Almost are based on intractability assumptions such as bilinear decision Diffie-Hellman exponent (BDDHE) [7,23], decisional bilinear Diffie-Hellman (DBDH) [11,13,35], and decisional linear (DLIN) [11]. Some of them are provably secure in the generic group model (GGM) [7], [23], while others – in the standard model (SDM) [35], [11], [13] .

Recall that lattice-based cryptosystems are believed to enjoy post-quantum (PQ) security. In contrast, cryptosystems, whose security is based on the intractability of factorization or discrete logarithm, are breakable by quantum adversaries [31]. With the rapid development of large-scale quantum computers [4], it is imperative to design cryptosystems which are secure against quantum adversaries. This leads us to the following question: "*Is it possible to design SE in the lattice setting?*". We found out that an answer to the question has already existed in the literature. One can get a lattice-based SE using a generic construction from the hierarchical inner product encryption (HIPE) given by Chen et al. in [12]. The generic construction deploys two lattice-based SE schemes from [1] and [34]. We base the security of both schemes on the intractability of

[4] For instance, see at https://www.nature.com/articles/d41586-019-03213-z

the learning with errors (LWE) problem. Unfortunately, such (lattice-based) SE construction is not free from a few weaknesses shown below.

Table 1: (D)SE constructions

| **Literature** | **From** | **Assumption** | **Security model** | **Selective** (S) /**Adaptive** (A) | **PQ security** |
|---|---|---|---|---|---|
| Boneh, Hamburg [7] | | BDDHE | GGM | S & A | ✗ |
| Hamburg [23] | | BDDHE | GGM | S | ✗ |
| Chen, Wee [13] | | DBDH | SDM | S | ✗ |
| Chen et al. [11] | | DLIN, DBDH | SDM | A | ✗ |
| Zhou, Cao [35] | | DBDH | SDM | S | ✗ |
| Abdalla et al. [1] | HIPE via Chen [12] | LWE | SDM | S | ✓ |
| Xagawa [34] | HIPE via Chen [12] | LWE | SDM | S | ✓ |
| **Our work** | DMIPE | LWE | SDM | S | ✓ |

**Shortcomings of Constructing SE from HIPE.** The notion of inner product encryption (IPE) was introduced by Katz, Sahai, and Waters [25]. Hierarchical inner product encryption (HIPE) is an extension of IPE introduced by Okamoto and Takashima [28]. A HIPE is identified by a field $\mathbb{F}$, a tuple $\Delta(\delta) := (\delta; \ell_1, \cdots, \ell_\delta)$ called *hierarchical format* of depth $\delta$. Specifically, for any $\vec{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_k) \in \Gamma_{|k} := \Gamma_1 \times \cdots \times \Gamma_k$, where $\Gamma_i := \mathbb{F}^{\ell_i}$, a hierarchical predicate $f_{\vec{V}}(\cdot)$ is defined as follows: $f_{\vec{V}}(\vec{X}) = 1$ for any $\vec{X} = (\mathbf{x}_1, \cdots, \mathbf{x}_t) \in \Gamma_{|t}$ if and only if $k \leq t$ and $\langle \mathbf{v}_i, \mathbf{x}_i \rangle = 0$, for all $i \in [k]$. Informally, for each pair $\vec{X}$ and $\vec{V}$, the criteria for successful decryption is that $f_{\vec{V}}(\vec{X})$ is equal to 1.

Chen et al. [12] have investigated the relation between SE and HIPE. They show that we can construct a $d$-dimensional linear SE from $\Delta(d) := (d; d, \cdots, d)$-HIPE (i.e., $\ell_1 = \cdots = \ell_d = d$) and vice versa (but the dimension of SE and the hierarchical format of HIPE might change). To construct SE from HIPE, the authors change the "belong to" relation into the "orthogonal to" relation, i.e. $\mathbf{x} \in V \Leftrightarrow \langle \mathbf{x}, \mathbf{v} \rangle = 0 \; \forall \mathbf{v} \in V^\perp$, where $V^\perp$ denotes the orthogonal complement of $V$. If we denote a basis of $V^\perp$ by $\mathcal{B}^\perp(V)$, then this is equivalent to $\langle \mathbf{x}, \mathbf{v} \rangle = 0$ for all $\mathbf{v}_i \in \mathcal{B}^\perp(V)$. In order to deploy HIPE for SE, we set $\vec{X} := (\mathbf{x}, \cdots, \mathbf{x})$ and $\vec{V} := \{\mathbf{v}_i : \mathbf{v}_i \in \mathcal{B}^\perp(V)\}$ for each $\mathbf{x}$ and $V$, respectively. Thanks to Linear Algebra (see Lemma 6 in Section 4) and the delegation capability of HIPE, one can perform delegation for SE.

The following shortcomings of the above construction can be identified:

- There is a single vector that is involved in SE encryption. Decryption keys may involve a list of vectors. In contrast, HIPE encryption takes many vectors. That is why in the construction of SE from HIPE, one has to duplicate the attribute vector of SE many times to fit the hierarchical format of HIPE.
- It is difficult, in general, to instantiate HIPE for practical use because of its complex structure. It is worth noting that there are only some other lattice-based HIPE constructions (for instance, these from Abdalla et al. [1] and Xagawa [34]). Unfortunately, they are not efficient enough.

The above considerations lead us to the following question: *"Can we construct SE from an IPE-related primitive that is simpler than HIPE?"* We give an affirmative answer to this question by introducing the new notion of delegatable multiple inner product encryption (DMIPE).

### 1.2 Contributions

Below we list the main results of our work.

- We introduce a novel primitive called delegatable multiple inner product encryption (DMIPE). It is a natural extension of IPE. We give a novel design of DMIPE using LWE. We prove that our design is selective payload-hiding secure in the standard model.
- We show an equivalence between DMIPE and SE, which provides a generic framework for constructing SE from DMIPE. As a result, we obtain a lattice-based SE, which is more efficient (in terms of sizes) than SEs constructed from HIPE. Conversely, we can also build DMIPE from SE. Moreover, the conversions between DMIPE and SE are security-preserving.
- We formally define the allow/deny-list encryption (ADE), which subsumes some other important primitives, e.g., PE [22], FuPE [15], DFPE [16]. We point out that one can build two versions of ADE from SE under appropriate embeddings.

### 1.3 Overview of Our Results.

**DMIPE.** The notion of DMIPE originates from IPE and is equipped with a delegation mechanism for producing decryption keys. In particular, a DMIPE ciphertext is connected to its *attribute vector*. A DMIPE decryption key can be generated from either the master secret key or from other secret keys by adding more vectors to the list of *predicate vectors*. There is an important requirement for predicate vectors, say $\vec{V}$. They have to be linearly independent, i.e. no vector is a linear combination of two or more other vectors from $\vec{V}$. The requirement is necessary to ensure that there is no redundant vector in $\vec{V}$ when checking decryption conditions. Besides, delegation of a decryption key for $\vec{V} \cup \mathbf{v}$ is possible if $\mathbf{v}$ is linearly independent from the existing predicate vectors in $\vec{V}$. Further details can be found in Section 3.

We show that DMIPE can be used to implement other primitives, e.g., SE, which can be exploited to build other primitives, as previously mentioned. We argue that DMIPE is a generalization of IPE, and it is more natural than HIPE. Compared to HIPE, the decryption hierarchy in DMIPE is more flexible for delegation. (See Table 2 for a quantitative comparison of IPE, HIPE and DMIPE and Figure 1 for an intuitive illustration of their relation). From the comparison in Table 2, it is easy to see that DMIPE and HIPE are not equivalent in the sense of transforming one into another. Details will be presented in Section 4.

**Lattice-based DMIPE.** At a high-level description, our lattice-based DMIPE design exploits the lattice trapdoor mechanism and the lattice evaluation for

Table 2: Comparison of IPE, HIPE and DMIPE.

| | **IPE** | **HIPE** | **DMIPE** |
|---|---|---|---|
| **# Attribute vectors** | 1 | $d$ | 1 |
| **# Predicate vectors** | 1 | $\leq d$ | $\geq 1$ |
| **Delegation?** | No | Yes | Yes |
| **Dimension of vectors** | same | not necessarily same | same |

inner product functions (see Lemma 4 and Lemma 5 for formal statements). The DMIPE design's security is based on the intractability of the decision Learning with Errors problem. The lattice-based DMIPE will be given in Section 5.

**Equivalence of DMIPE and SE.** We prove in Section 6 that DMIPE and SE are equivalent in the sense that we can establish *security-preserving* conversions between them. In particular, we can use DMIPE to construct SE, where SE inherits security from DMIPE and vice versa. It also means we can get a lattice-based SE from a lattice-based DMIPE. This way, our ($d$-dimensional) SE construction is more efficient in terms of sizes, than SE obtained from $\Delta(d)$-HIPEs [1, 34], according to the generic framework of Chen et al. [12]. Table 3 compares different lattice-based SEs.

Table 3: Comparison of lattice-based $d$-dimensional SEs

| **$d$-dim. SE from** | **pk-size** ($\ell := \lceil \log_r q \rceil$) | **msk-size** ($\ell := \lceil \log_r q \rceil$) | **sk-size** ( $k$ predicates) | **ct-size** ($h$ attributes, $m$-bit message) |
|---|---|---|---|---|
| Abdalla et al. [1] ($\Delta(d)$-HIPE) | $d^2(\ell+1)\cdot\mathbb{Z}_q^{n\times m}$ $+2\cdot\mathbb{Z}_q^{n\times m}$ | $1\cdot D_{\mathbb{Z}}^{m\times m}$ | $1\cdot D_{\mathbb{Z}}^{km\times m}$ | $(hd(\ell+1))\cdot\mathbb{Z}_q^{m}$ $+2\cdot\mathbb{Z}_q^{m}$ |
| Xagawa [34] ($\Delta(d)$-HIPE) | $(d^2+d)\cdot\mathbb{Z}_q^{n\times n\ell}$ $+2\cdot\mathbb{Z}_q^{n\times ml}$ | $1\cdot D_{\mathbb{Z}}^{(m-n\ell)\times n\ell}$ | $1\cdot D_{\mathbb{Z}}^{(m+(2k-1)n\ell)\times m}$ $+1\cdot D_{\mathbb{Z}}^{(m+(2k-1)n\ell)\times nk}$ | $(h-1+hd)\cdot\mathbb{Z}_q^{n\ell}$ $+2\cdot\mathbb{Z}_q^{m}$ |
| Ours (DMIPE) | $(d+2)\cdot\mathbb{Z}_q^{n\times m}$ | $1\cdot D_{\mathbb{Z}}^{m\times m}$ | $1\cdot D_{\mathbb{Z}}^{km\times m}$ | $(d+2)\cdot\mathbb{Z}_q^{m}$ |

**ADE and the Construction from SE.** ADE is, in fact, also a subclass of PrE, in which both predicates and attributes are *tags*. These tags are categorized into two lists: *allow list* contains positive tags and *deny list*–negative tags. Both ciphertexts and decryption keys are associated with these two kinds of tags. Further, ADE also supports the delegation mechanism, which is called *puncturing*. Roughly saying, *negatively puncturing* is the delegation on negative tags, and this puncturing can revoke the decryption ability. In contrast, *positively puncturing* is delegation done on positive tags and allows decryption. We will present a formal definition and security model for ADE. Moreover, we consider three versions of ADE: (i) standard ADE (sADE); (ii) inclusive ADE (iADE) and (iii) $k$-threshold ADE ($k$-tADE). We show that one can construct sADE and iADE from SE by applying some appropriate encodings. This result will be detailed in Section 7. However, how to translate $k$-tADE to SE is an open problem.

The security notions for SE, DMIPE and even ADE inherit from those for PrE, which were introduced in [25]. They include selective payload-hiding, selective attribute-hiding, adaptive payload-hiding, and adaptive attribute-hiding. We stress that, in this work, we concentrate on the selective payload-hiding security.

## 2 Preliminaries

Given a positive integer $n$, $[n]$ stands for the set $\{1, \cdots, n\}$. We write $W \sqsubseteq V$ for the fact that $W$ is an (affine or vector) subspace of $V$. The notation $\mathbf{A} \otimes \mathbf{B}$ is a tensor product of two matrices $\mathbf{A}$ and $\mathbf{B}$. Throughout this work, we represent a vector with a small bold-face letter, e.g., $\mathbf{x}$ and in the column form unless stated otherwise. We write a matrix in capital bold-face, e.g., $\mathbf{B}$. We write $\mathbf{b}^\top$ (resp., $\mathbf{A}^\top$) to denote the transpose of a vector $\mathbf{b}$ (resp., a matrix $\mathbf{A}$). The notation $\widetilde{\mathbf{S}} := [\widetilde{\mathbf{s}}_1 | \cdots | \widetilde{\mathbf{s}}_k]$ stands for the Gram-Schmidt (GS) orthogonalisation of $\mathbf{S} := [\mathbf{s}_1 | \cdots | \mathbf{s}_k]$. The notation $U(X)$ means the uniform distribution over the set $X$. All logarithms are for base 2. All norms are the max-absolute-value norm $\| \cdot \|_{\max}$[5] unless otherwise stated. The norm returns the maximum absolute value of the entries of an input vector/matrix. For example, for a vector $\mathbf{a} = (a_1, \cdots, a_n)$ and a matrix $\mathbf{A} = (a_{i,j})_{i \in [n]}^{j \in [m]}$, $\|\mathbf{a}\|_{\max} := \max_{i \in [n]} |a_i|$ and $\|\mathbf{A}\|_{\max} := \max_{i \in [n], j \in [m]} |a_{i,j}|$. The following lemma is the well-known result regarding the max-absolute-value norm.

**Lemma 1.** *Let $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ be vectors of dimensions $m_1, m_2, m_3 \in \mathbb{N}$, respectively. Let $\mathbf{A}_1, \mathbf{A}_2$ be matrices of appropriate dimensions. Then, $\|\mathbf{e}_1^\top \mathbf{A}_1\|_{\max} \leq m_1 \|\mathbf{e}_1^\top\|_{\max} \cdot \|\mathbf{A}_1\|_{\max}$, and $\|(\mathbf{e}_2^\top | \mathbf{e}_3^\top)\mathbf{A}_2\|_{\max} \leq (m_2 \|\mathbf{e}_2^\top\|_{\max} + m_3 \|\mathbf{e}_3^\top\|_{\max}) \cdot \|\mathbf{A}_2\|_{\max}$.*

### 2.1 Framework of Spatial Encryption

**Syntax.** Formally, SE [7,23] consists of five main algorithms $\mathsf{SE.Setup}$, $\mathsf{SE.Derive}$, $\mathsf{SE.Del}$, $\mathsf{SE.Enc}$, $\mathsf{SE.Dec}$ described as follows:

$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{SE.Setup}(1^\lambda, \mathsf{sp})$**:** The algorithm takes as input a security parameter $\lambda$ and setup parameters $\mathsf{sp}$. It returns public parameters $\mathsf{pp}$ which implicitly defined a top space $\mathsf{T}$ and a master secret key $\mathsf{msk}$. The master key $\mathsf{msk}$ can be seen as the secret key $\mathsf{sk}_\mathsf{T}$ (i.e., $\mathsf{msk} = \mathsf{sk}_\mathsf{T}$) for the top space $\mathsf{T}$.

$\mathsf{sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$**:** The algorithm takes as input the master secret key $\mathsf{msk}$ and a subspace $V$. It outputs the secret key $\mathsf{sk}_V$ for $V$.

$\mathsf{sk}_{V_2} \leftarrow \mathsf{SE.Del}(\mathsf{pp}, \mathsf{sk}_{V_1}, V_2)$**:** The algorithm takes as input the secret key $\mathsf{sk}_{V_1}$ for the space $V_1$. It outputs the secret key $\mathsf{sk}_{V_2}$ for $V_2$, where $V_2 \sqsubseteq V_1$.

$\mathsf{ct}_\mathbf{x} \leftarrow \mathsf{SE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$**:** The encryption algorithm encrypts a message $\mu$ under a point/vector $\mathbf{x}$. It outputs a ciphertext $\mathsf{ct}_\mathbf{x}$.

[5] Some papers (e.g., [9], [33], [24]) denote this max-absolute-value norm by $\| \cdot \|_\infty$.

$\mu/\perp \leftarrow \mathbf{SE.Dec}(\mathbf{pp}, \mathbf{ct_x}, \mathbf{sk}_V)$**:** The decryption algorithm takes as input a secret key $\mathsf{sk}_V$ and a ciphertext $\mathsf{ct}_{\mathbf{x}}$. Decryption succeeds if $\mathbf{x} \in V$ and it outputs the plaintext $\mu$. Otherwise, it fails and returns $\perp$.

| |
|---|
| GAME $\mathsf{SE}^{\mathsf{sel,ATK}}_{\mathsf{payload},\mathcal{A}}(\lambda, \mathsf{sp})$ : <br> (where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$) <br> 1. $\mathbf{x}^* \leftarrow \mathcal{A}(1^\lambda, \mathsf{sp})$; 2. $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{SE.Setup}(1^\lambda, \mathsf{sp})$; <br> 3. $(\mu_0^*, \mu_1^*) \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\ \mathsf{DQ}_1(\cdot,\cdot)}(\mathsf{pp})$; 4. $b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}^*_{\mathbf{x}^*} \leftarrow \mathsf{SE.Enc}(\mathsf{pp}, \mathbf{x}^*, \mu_b^*)$; <br> 5. $b' \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\ \mathsf{DQ}_2(\cdot,\cdot)}(\mathsf{pp}, \mathsf{ct}^*_{\mathbf{x}^*})$. //**NOTE:** $\mathsf{DQ}_2(V, \mathsf{ct}^*_{\mathbf{x}^*})$ is not allowed with $\mathbf{x}^* \in V$. <br> 6. If $b' = b$, return 1. Otherwise, return 0. |
| **Queried Oracles:** <br> • Key Oracle $\mathsf{KQ}(\mathrm{V})$ (allowed only if $\mathbf{x}^* \notin V$): Return $\mathsf{sk}_V \leftarrow \mathsf{SE.Der}(\mathsf{pp}, \mathsf{msk}, V)$. <br> • Decryption Oracle $\mathsf{DQ}_1(V, \mathsf{ct}_{\mathbf{x}})$ (allowed only if $\mathsf{ATK} \in \{\mathsf{CCA1}, \mathsf{CCA2}\}$): Run $\mathsf{sk}_V \leftarrow \mathsf{SE.Der}(\mathsf{pp}, \mathsf{msk}, V)$, then return the output of $\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$. <br> • Decryption Oracle $\mathsf{DQ}_2(V, \mathsf{ct}_{\mathbf{x}})$ (allowed only if $\mathsf{ATK} = \mathsf{CCA2}$): Run $\mathsf{sk}_V \leftarrow \mathsf{SE.Der}(\mathsf{pp}, \mathsf{msk}, V)$, then return the output of $\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$. |

Fig. 2: Selective payload-hiding security game for SE

**Correctness.** It requires that for all $\lambda, \mathsf{sp}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{SE.Setup}(1^\lambda, \mathsf{sp}))$, $\mathsf{ct}_{\mathbf{x}} \leftarrow \mathsf{SE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$, $\mathsf{sk}_V \leftarrow \mathsf{SE.Del}(\mathsf{pp}, \mathsf{sk}_{V'}, V)$ (for some $V'$ such that $V \sqsubseteq V'$) or $\mathsf{sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk})$, if $\mathbf{x} \in V$ then $\Pr[\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{sk}_V, \mathsf{ct}_{\mathbf{x}}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$; otherwise, $\Pr[\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{sk}_V, \mathsf{ct}_{\mathbf{x}}) = \mu] < \mathsf{negl}(\lambda)$.

We also require that the distribution of secret keys $\mathsf{sk}_V$ for any subspace $V$ must be the same. It should depend neither on how a key is produced (i.e. by either $\mathsf{SE.Derive}$ or $\mathsf{SE.Del}$) nor on what a path is (e.g., the direct path from the top space or the path of a delegation from another subspace).

**Security Notions for SE.** SE security notions include selective/adaptive payload/ attribute-hiding. However, we only formally define the selective payload-hiding security for SE – see Definition 1 and the game $\mathsf{SE}^{\mathsf{sel,\ ATK}}_{\mathsf{payload},\mathcal{A}}$ in Figure 2. Note that $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, where $\mathsf{CPA}$ and $\mathsf{CCA}$ stand for chosen plaintext and chosen ciphertext attacks, respectively.

**Definition 1 (Selective Payload-hiding Security for SE).** *SE is selective payload-hiding secure if the advantage of the adversary playing in the* $\mathsf{SE}^{\mathsf{sel,ATK}}_{\mathsf{payload},\mathcal{A}}$ *game is negligible, i.e.,* $\mathsf{Adv}^{\mathsf{payload,ATK}}_{\mathsf{SE},\mathcal{A},\mathsf{sel}} := |\Pr[b' = b] - 1/2| = \mathsf{negl}(\lambda)$.

### 2.2 Lattices, Gaussians, Trapdoors, Lattice Evaluations for Inner Product Functions

We focus on the following *lattices*: $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{Ae} = 0 \pmod q\}$, $\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m | \mathbf{Ae} = \mathbf{u} \pmod q\}$, $\Lambda_q^{\mathbf{U}}(\mathbf{A}) := \{\mathbf{R} \in \mathbb{Z}^{m \times k} | \mathbf{AR} = \mathbf{U} \pmod q\}$. Note that $\mathbb{Z}^m$ for any $m \in \mathbb{N}$ is also a lattice.

We also involve with a *discrete Gaussian distribution* over $\mathcal{L}$ centred at $\mathbf{v}$ with (Gaussian) parameter $\sigma$ which is defined by $D_{\mathcal{L},\sigma,\mathbf{v}}(\mathbf{x}) = \frac{\rho_{\sigma,\mathbf{v}}(\mathbf{x})}{\rho_{\sigma,\mathbf{v}}(\mathcal{L})}$ for all $\mathbf{x} \in \mathcal{L}$, where $\rho_{\sigma,\mathbf{v}}(\mathbf{x}) = \exp(-\pi\|\mathbf{x}-\mathbf{v}\|^2/\sigma^2)$ and $\rho_{\sigma,\mathbf{v}}(\mathcal{L}) := \sum_{\mathbf{x}\in\mathcal{L}} \rho_{\sigma,\mathbf{v}}(\mathbf{x})$. In case $\mathbf{v} = \mathbf{0}$, we just write $D_{\mathcal{L},\sigma}$. We also consider the $(B,\epsilon)$-*bounded distributions* $\chi$ supported over $\mathbb{Z}$. The following lemma says how short a vector sampled via a discrete Gaussian distribution (over $\mathbb{Z}$) is.

**Lemma 2 ( [26, Lemma 4.4]).** $\Pr[|x| > k\sigma : \ x \leftarrow D_{\mathbb{Z},\sigma}] \leq 2\exp(-\frac{k^2}{2})$.

Note that, in Lemma 2, if we set $k = 12$, then $\Pr[|x| \leq 12\sigma : \ x \leftarrow D_{\mathbb{Z},\sigma}] \geq 1 - 2\exp(-72) \approx 1 - 2^{-100}$. Then, $\chi = D_{\mathbb{Z},\sigma}$ is a $(12\sigma, 2^{-100})$-bounded distribution.

The following leftover hash lemma enables us to replace a random matrix by a pseudo-random one in hybrid games for our security proofs.

**Lemma 3 (Leftover Hash Lemma, [2, Lemma 4]).** *Given $m, n, q$ are positive integers such that $m > (n+1)\log q + \omega(\log n)$, $k = \mathsf{poly}(n)$, $q > 2$ is a prime, and that $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$ and $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n\times k}$. Then, the joint distributions $(\mathbf{A}, \mathbf{AR}, \mathbf{e}^\top\mathbf{R})$ and $(\mathbf{A}, \mathbf{B}, \mathbf{e}^\top\mathbf{R})$ are statistically close to each other for any matrix $\mathbf{R} \xleftarrow{\$} \{-1,0,1\}^{m\times k}$ and for all vectors $\mathbf{e} \in \mathbb{Z}_q^m$.*

The Decisional Learning with Errors (DLWE) [29] is used as our intractability assumption to prove the security of our DMIPE design. The hardness of DLWE can be found, e.g., in [9, Corollary 3].

We follow the works [9], [33], [24] for *lattice trapdoors*. In particular, it is shown in [27] that the gadget matrix $\mathbf{G}$ has a publicly known constant trapdoor denoted in this work as $\mathbf{G}^{-1}_{O(1)}$.

**Lemma 4 ( [4], [18], [2], [10], [27]).** *The following facts hold for lattice trapdoors:*

1. *Let $n, m, q$ be positive integers where $m = O(n\log q)$. There is an efficient algorithm* TrapGen *that takes $(n, m, q)$ as input to generate a matrix $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ together with its trapdoor $\mathbf{A}^{-1}_{\sigma_0}$ satisfying that $\mathbf{A} \overset{negl}{\sim} U(\mathbb{Z}_q^{n\times m})$ with $\sigma_0 = \omega(n\log q\log n)$.*
2. *Given a trapdoor $\mathbf{A}^{-1}_{\sigma_1}$, one can compute $\mathbf{A}^{-1}_{\sigma_2}$ for any $\sigma_2 \geq \sigma_1$.*
3. *Given a trapdoor $\mathbf{A}^{-1}_{\sigma}$, one can compute $[\mathbf{A}|\mathbf{B}]^{-1}_{\sigma}$, $[\mathbf{B}|\mathbf{A}]^{-1}_{\sigma}$ for any matrix $\mathbf{B}$ having the same number of rows as $\mathbf{A}$.*
4. *Given the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n\times m'}$ with $m' \geq n\lceil\log q\rceil$, using its trapdoor $\mathbf{G}^{-1}_{O(1)}$ one can compute the trapdoor $[\mathbf{A}|\mathbf{AR}+\mathbf{G}]^{-1}_{\sigma}$ for all $\mathbf{A} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{R} \in \mathbb{Z}^{m\times m'}$ and $\sigma = m \cdot \|\mathbf{R}\|_{\max} \cdot \omega(\sqrt{\log m})$.*
5. *For a trapdoor $\mathbf{A}^{-1}_{\sigma_1}$ and for any $\mathbf{U} \in \mathbb{Z}_q^{n\times m'}$, by Lemma 2, $\Pr[\|\mathbf{A}^{-1}_{\sigma_1}(\mathbf{U})\|_{\max} \leq 12\sigma : \ \mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}] \geq 1 - 2^{-100}$.*

For every $\mathbf{x} \in \mathbb{Z}_q^d$, an inner product function $f_{\mathbf{v}} : \mathbb{Z}_q^d \to \mathbb{Z}_q$ is indexed by a vector $\mathbf{v} \in \mathbb{Z}_q^d$ and is defined as $f_{\mathbf{v}}(\mathbf{x}) := \langle \mathbf{v}, \mathbf{x} \rangle \pmod q$. Thus, [33, Theorem 2], for example, is not suitable for our work. The following lemma is sufficient for our purpose.

**Lemma 5 (Evaluation for Inner Product Functions).** *There exist an efficient deterministic algorithm* $\mathsf{EvalF}^{\mathsf{IP}}$ *such that for all* $n, q, d \in \mathbb{N}$ *and* $m = n\lceil \log q \rceil$*, for any inner product function* $f_{\mathbf{v}} : \mathbb{Z}_q^d \to \mathbb{Z}_q$ *indicated by* $\mathbf{v} \in \mathbb{Z}_q^d$*, and for any matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times md}$*, it outputs a matrix* $\mathbf{H} \in \{0,1\}^{md \times m} \leftarrow \mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}}, \mathbf{B})$*, satisfying that* $\|\mathbf{H}\|_{\max} \leq 1$ *and that for every* $\mathbf{x} \in \mathbb{Z}_q^d$*,*

$$[\mathbf{B} \pm \mathbf{x} \otimes \mathbf{G}]\mathbf{H} = \mathbf{B}\mathbf{H} \pm \langle \mathbf{v}, \mathbf{x} \rangle \cdot \mathbf{G} \pmod q.$$

*Proof.* See Appendix A. □

## 3 Delegatable Multiple Inner Product Encryption

In this section, we present the syntax and security notions for DMIPE. For DMIPE, a ciphertext is produced together with a $d$-dimensional vector. We call it *ciphertext vector*, or *attribute vector*. A secret key contains a list of one or multiple vectors of dimension $d$. We call them *key vectors* or *predicate vectors*. All vectors are supposed to belong to the same domain (space) $\mathcal{D}$. The domain supports typical or symbolic inner product operations. The operation is defined as $\langle \mathbf{x}, \mathbf{v} \rangle = x_1 v_1 + \cdots x_d v_d \in \mathcal{D}$ for $\mathbf{x} := (x_1, \cdots, x_d)$, $\mathbf{v} := (v_1, \cdots, v_d) \in \mathcal{D}^d$. Note that $\mathcal{D}$ can be $\mathbb{Z}$ or even $\mathbb{Z}_q$.

**Syntax of DMIPE**. A DMIPE consists of the five algorithms $\mathsf{DMIPE.Setup}$, $\mathsf{DMIPE.Derive}$, $\mathsf{DMIPE.Del}$, $\mathsf{DMIPE.Enc}$ and $\mathsf{DMIPE.Dec}$. They are formally defined below.

- $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$: The algorithm takes as input a security parameter $\lambda$ and setup parameters $\mathsf{sp}$. It returns public parameters $\mathsf{pp}$ and a master secret key $\mathsf{msk}$.
- $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$: The algorithm takes a master secret key $\mathsf{msk}$ and a list of vector $\vec{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$. It returns a secret key $\mathsf{sk}_{\vec{V}}$ for $\vec{V}$.
- $\perp / \mathsf{sk}_{\vec{V}_2} \leftarrow \mathsf{DMIPE.Del}(pp, \mathsf{sk}_{\vec{V}_1}, \mathbf{v}_{k+1})$: The algorithm takes the secret key $\mathsf{sk}_{\vec{V}_1}$ for $\vec{V}_1 = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$ and returns a secret key $\mathsf{sk}_{\vec{V}_2}$ for $\vec{V}_2 := \vec{V}_1 \cup \{\mathbf{v}_{k+1}\}$. If $\mathbf{v}_{k+1}$ is not linearly independent of $\vec{V}$, then it returns $\perp$.
- $\mathsf{ct}_{\mathbf{x}} \leftarrow \mathsf{DMIPE.Enc}(pp, \mu, \mathbf{x})$: The algorithm encrypts a message $\mu$ under a vector $\mathbf{x}$ and produces a ciphertext $ct_{\mathbf{x}}$.
- $\perp / \mu \leftarrow \mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathsf{ct}_{\mathbf{x}})$: The algorithm decrypts a ciphertext $\mathsf{ct}_{\mathbf{x}}$ using a secret key $\mathsf{sk}_{\vec{V}}$. It is successful if $\vec{V} \cdot \mathbf{x} = \mathbf{0}$ (i.e., $\langle \mathbf{v}_i, \mathbf{x} \rangle = 0$ for all $\mathbf{v}_i \in \vec{V}$). If the condition does not hold, it fails and returns $\perp$.

**Correctness of DMIPE.** For all $\lambda, \mathsf{sp}$, $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$, $\mathsf{ct}_{\mathbf{x}} \leftarrow \mathsf{DMIPE.Enc}(\mathsf{pp}, \mu, \mathbf{x})$, $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Del}(\mathsf{pp}, \mathsf{sk}_{\vec{V}'}, \mathbf{v})$ (where $\vec{V} = \vec{V}' \cup \{\mathbf{v}\}$)

or $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$, if $\vec{V} \cdot \mathbf{x} = \mathbf{0}$ then $\Pr[\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathsf{ct}_{\mathbf{x}}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$; otherwise, $\Pr[\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathsf{ct}_{\mathbf{x}}) = \mu] < \mathsf{negl}(\lambda)$.

**Security Notions of DMIPE.** Same as SE, we only consider selective payload-hiding security for DMIPE. Definiton 2 and Figure 3 together describe our security notion.

**Definition 2.** *DMIPE is selective payload-hiding secure if the advantage of the adversary playing in the* $\mathsf{DMIPE}^{\mathsf{sel},\mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}$ *game (in Figure 3) is negligible, i.e.,* $\mathsf{Adv}^{\mathsf{payload},\mathsf{ATK}}_{\mathsf{DMIPE},\mathcal{A},\mathsf{sel}} := |\Pr[b' = b] - 1/2| = \mathsf{negl}(\lambda)$.

GAME $\mathsf{DMIPE}^{\mathsf{sel},\mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}(\lambda, \mathsf{sp})$:
(where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$)
1. $\mathbf{x}^* \leftarrow \mathcal{A}(1^\lambda, \mathsf{sp})$; 2. $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$;
3. $(\mu_0^*, \mu_1^*) \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\ \mathsf{DQ}_1(\cdot,\cdot)}(\mathsf{pp})$; 4. $b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}^*_{\mathbf{x}^*} \leftarrow \mathsf{DMIPE.Enc}(\mathsf{pp}, \mathbf{x}^*, \mu_b^*)$;
5. $b' \leftarrow \mathcal{A}^{\mathsf{ct}^*_{\mathbf{x}^*},\ \mathsf{KQ}(\cdot),\ \mathsf{DQ}_2(\cdot,\cdot)}(\mathsf{pp})$. // **NOTE:** $\mathsf{DQ}_2(\vec{V}, \mathsf{ct}^*_{\mathbf{x}^*})$ with $\mathbf{x}^* \perp \vec{V}$ is not allowed.
6. If $b' = b$, return 1. Otherwise, return 0.

**Queried Oracles:**
- $\mathsf{KQ}(\vec{V})$ (allowed only if $\mathbf{x}^* \not\perp \vec{V}$): Run $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$.
- $\mathsf{DQ}_1(\vec{V}, \mathsf{ct}_{\mathbf{x}})$ (allowed only if $\mathsf{ATK} \in \{\mathsf{CCA1}, \mathsf{CCA2}\}$): Run $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$, then return the output of $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_{\vec{V}})$.
- $\mathsf{DQ}_2(\vec{V}, \mathsf{ct}_{\mathbf{x}})$ (allowed only if $\mathsf{ATK} = \mathsf{CCA2}$): Run $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$, then return the output of $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_{\vec{V}})$.

Fig. 3: Selective payload-hiding security game for DMIPE. Here if $\mathbf{x} \perp \mathbf{v}, \forall \mathbf{v} \in \vec{V}$ then we write $\mathbf{x} \perp \vec{V}$; otherwise we write $\mathbf{x} \not\perp \vec{V}$.

## 4 Generic SE Construction from DMIPE

We only focus on a linear SE, where components are vectors and vector subspaces over some field $\mathbb{F}$, e.g., $\mathbb{F} = \mathbb{Z}_q$ for $q$ prime. Note that we can always embed a $d$-dimensional affine SE into a $(d+1)$-dimensional linear SE, as shown below. First, we recap some notions in affine/linear algebra.

Let $\mathbb{F}$ be a field. A $d$-dimensional vector subspace $V \sqsubseteq \mathbb{F}^d$ can be represented as $V := \mathsf{span}(\mathbf{M}) = \{\mathbf{M}\mathbf{x} : \mathbf{x} \in \mathbb{F}^m\}$ for some $\mathbf{x} \in \mathbb{F}^m$, where $\mathbf{M} \in \mathbb{F}^{d \times m}$ is a basis for $V$. Note that all rows of $\mathbf{M}$ are linearly independent. A $d$-dimensional affine subspace $W$ of $\mathbb{F}^d$ can be represented as $W = \mathbf{y} + \mathsf{span}(\mathbf{M}) = \{\mathbf{y} + \mathbf{M}\mathbf{x} : \mathbf{x} \in \mathbb{F}^m\}$ for some $\mathbf{y} \in \mathbb{F}^d, \mathbf{M} \in \mathbb{F}^{d \times m}$. We can transform $W$ to a vector subspace defined as $W = \mathsf{span}(\mathbf{M}') := \left\{\mathbf{M}'\mathbf{x}' : \mathbf{x}' = \begin{pmatrix}1\\ \mathbf{x}\end{pmatrix}, \mathbf{x} \in \mathbb{F}^{m+1}\right\}$, where $\mathbf{M}'$ has the form $\begin{bmatrix}1 & 0\\ \mathbf{y} & \mathbf{M}\end{bmatrix} \in \mathbb{F}^{(d+1)\times(m+1)}$. Obviously, all rows of $\mathbf{M}'$ are still linearly independent assuming the linear independence for $\mathbf{M}$'s rows. Then $W$ now is a vector subspace of dimension $d+1$. For linear SE, recall that we encrypt

a plaintext together with a vector $\mathbf{x}$ and a decryption key is produced using a vector space $V$. Successful decryption using the decryption key requires that $\mathbf{x} \in V$. We need a tool that helps us transform the "belong to" relation for the SE syntax to the "orthogonal to" relation compatible with the DMIPE syntax. The following well-known lemma from Linear Algebra helps us to compute the basis for the orthogonal complement of a vector space.

**Lemma 6 ( [14, Algorithm 2.3.7] and [12]).** *There exists an efficient algorithm, named* $\mathsf{OCB}$*, such that on input a vector space* $V$*, outputs a basis, called* $\mathcal{B}^{\perp}(V)$*, for the orthogonal complement* $V^{\perp}$ *of* $V$*. Furthermore, the algorithm guarantees that if* $V_2 \sqsubseteq V_1$ *then* $\mathcal{B}^{\perp}(V_1) \subseteq \mathcal{B}^{\perp}(V_2)$*.*

Now we are ready to present our generic SE construction from DMIPE. Given a DMIPE scheme $\Pi_{\mathsf{DMIPE}} := (\mathsf{DMIPE.Setup}, \mathsf{DMIPE.Derive}, \mathsf{DMIPE.Del}, \mathsf{DMIPE.Enc}, \mathsf{DMIPE.Dec})$. Then we can construct an SE scheme $\Pi_{\mathsf{SE}} := (\mathsf{SE.Setup}, \mathsf{SE.Derive}, \mathsf{SE.Del}, \mathsf{SE.Enc}, \mathsf{SE.Dec})$ as follows:

- $\mathsf{SE.Setup}(1^{\lambda}, \mathsf{sp})$: For input a security parameter $\lambda$, a system parameters $\mathsf{sp}$, run $(\mathsf{dmipe.pp}, \mathsf{dmipe.msk}) \leftarrow \mathsf{DMIPE.Setup}(1^{\lambda}, \mathsf{sp})$ and set $\mathsf{pp} := \mathsf{dmipe.pp}$, and $\mathsf{msk} := \mathsf{dmipe.msk}$.
- $\mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$: For input public parameters $\mathsf{pp}$, the master secret key $\mathsf{msk}$ and a subspace $V$, perform: Run $\mathcal{B}^{\perp}(V) \leftarrow \mathsf{OCB}(V)$, and set $\vec{V} := \{\mathbf{v} : \mathbf{v} \in \mathcal{B}^{\perp}(V)\}$. Run $\mathsf{dmipe.sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$, and set $\mathsf{sk}_V := \mathsf{dmipe.sk}_{\vec{V}}$.
- $\mathsf{SE.Del}(\mathsf{pp}, \mathsf{sk}_{V_1}, V_2)$: For input public parameters $\mathsf{pp}$, secret key for subspace $\mathsf{sk}_{V_1} = \mathsf{dmipe.sk}_{\vec{V}}$ for $V_1$ and a subspace $V_2 \sqsubseteq V_1$, perform: Run $\mathcal{B}^{\perp}(V_1) \leftarrow \mathsf{OCB}(V_1)$, $\mathcal{B}^{\perp}(V_2) \leftarrow \mathsf{OCB}(V_2)$, and set $\vec{V}_1 := \{\mathbf{v} : \mathbf{v} \in \mathcal{B}^{\perp}(V_1)\}$, $\vec{V}_2 := \{\mathbf{v} : \mathbf{v} \in \mathcal{B}^{\perp}(V_2)\}$. Note that, since $V_2 \sqsubseteq V_1$ , $\vec{V}_1 \subseteq \vec{V}_2$. Suppose that $\vec{V}_2 \setminus \vec{V}_1 = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$ for some $k \geq 1$. Set $\vec{V} \leftarrow \vec{V}_1$. For $i \in [k]$, run $\mathsf{dmipe.sk}_{\vec{V} \cup \{\mathbf{v}_i\}} \leftarrow \mathsf{DMIPE.Del}(\mathsf{pp}, \mathsf{dmipe.sk}_{\vec{V}}, \mathbf{v}_i)$, then set $\vec{V} \leftarrow \vec{V} \cup \{\mathbf{v}_i\}$. At this point, we reach $\vec{V} = \vec{V}_2$. Finally, output $\mathsf{sk}_{V_2} := \mathsf{dmipe.sk}_{\vec{V}_2}$. Doing this makes it clear that the distribution of the private keys is independent of the path taken. Namely, the distribution for the key $\mathsf{sk}_{V_3}$ computed from $\mathsf{sk}_{V_2}$ is the same as that of $\mathsf{sk}_{V_3}$ computed from $\mathsf{sk}_{V_1}$ with $V_3 \sqsubseteq V_2 \sqsubseteq V_1$.
- $\mathsf{SE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$: For input the public parameters $\mathsf{pp}$, an attribute vector $\mathbf{x}$ and a plaintext $\mu$, run $\mathsf{dmipe.ct}_{\mathbf{x}} \leftarrow \mathsf{DMIPE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$ and output a ciphertext $\mathsf{ct}_{\mathbf{x}} := \mathsf{dmipe.ct}_{\mathbf{x}}$.
- $\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$: On input the public parameters $\mathsf{pp}$, a ciphertext $\mathsf{ct}_{\mathbf{x}}$ and a secret key $\mathsf{sk}_V$ for a space $V$, return the output of $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$.

We establish the correctness of SE in Theorem 1.

**Theorem 1.** *The SE* $\Pi_{\mathsf{SE}}$ *is correct assuming correctness of the underlying DMIPE* $\Pi_{\mathsf{DMIPE}}$*.*

*Proof.* The correctness of $\Pi_{\mathsf{SE}}$ follows from the equivalence of the statements that "$\mathbf{x} \in V$" and that "$\mathbf{x} \perp \mathbf{v}$, for all $\mathbf{v} \in \mathcal{B}^{\perp}(V)$". □

**Theorem 2.** *Given an adversary $\mathcal{S}$ that plays against some security game (selective/adaptive payload-/attribute-hiding) for $\Pi_{\mathsf{SE}}$, one can build an adversary $\mathcal{A}$ playing against the same security game for $\Pi_{\mathsf{DMIPE}}$ such that* $\mathsf{Adv}^{\mathsf{DMIPE}}_{\mathcal{A}} \geq \mathsf{Adv}^{\mathsf{SE}}_{\mathcal{S}}$.

*Proof.* See Appendix A. □

## 5 Lattice-based DMIPE construction

For a vector $\mathbf{v} \in \mathbb{Z}_q^d$, we define an inner product function $f_{\mathbf{v}} : \mathbb{Z}_q^d \to \mathbb{Z}_q$ as $f_{\mathbf{v}}(\mathbf{x}) := \langle \mathbf{v}, \mathbf{x} \rangle \pmod q$, for any $\mathbf{x} \in \mathbb{Z}_q^d$. Recall that we can represent this function as an addition gate; see [6, Section 4]. Our lattice-based DMIPE construction exploits the lattice trapdoor mechanism [18] [2] [27] and the lattice evaluation algorithms developed in a long series of works [27], [19], [6], [33].

**The Construction.** The lattice-based DMIPE is presented right below.

- $\mathsf{DMIPE.Setup}(1^\lambda, 1^d)$: On input a security parameter $\lambda$, a dimension $d$, do the following: Choose $n, m, q$ according to $\lambda, d$. Also, choose a $(B, \epsilon)$-bounded distribution $\chi$ for the underlying LWE problem. We can take $\chi = D_{\mathbb{Z},\sigma^*}$ (for some $\sigma^* > 0$) which is a $(12\sigma^*, 2^{-100})$-bounded distribution. Choose a Gaussian parameter $\sigma_0$, and sample $(\mathbf{A}, \mathbf{A}^{-1}_{\sigma_0}) \leftarrow \mathsf{TrapGen}(n, m, q)$, $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times md}$. Output public parameters $\mathsf{pp} := (\mathbf{A}, \mathbf{B}, \mathbf{U})$ and master secret key $\mathsf{msk} := \mathbf{A}^{-1}_{\sigma_0}$.
- $\mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$: Taking as input public parameters $\mathsf{pp}$, a master secret key $\mathsf{msk}$ and a list of $d$-dimensional vectors $\vec{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$, perform: For each vector $\mathbf{v}_i$, evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}(f_{\mathbf{v}_i}, \mathbf{B})$ and compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$. Set $\mathbf{B}_{\vec{V}} := [\mathbf{B}_{\mathbf{v}_1} | \cdots | \mathbf{B}_{\mathbf{v}_k}]$ and $\mathbf{A}_{\vec{V}} := [\mathbf{A} | \mathbf{B}_{\vec{V}}]$. Compute trapdoor $\mathbf{A}^{-1}_{\vec{V},\sigma_0}$ for $\mathbf{A}_{\vec{V}}$ (via Item 3 of Lemma 4) and output $\mathsf{sk}_{\vec{V}} := \mathbf{A}^{-1}_{\vec{V},\sigma_0}$.
- $\mathsf{DMIPE.Del}(\mathsf{pp}, \mathsf{sk}_{\vec{V}_1}, \mathbf{v}_{k+1})$: On input public parameters $\mathsf{pp}$, a secret key $\mathsf{sk}_{\vec{V}_1} = \mathbf{A}^{-1}_{\vec{V}_1,\sigma_0}$ for a list $\vec{V}_1 = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$, and a vector $\mathbf{v}_{k+1} \notin \vec{V}_1$), do the following: For all $i \in [k+1]$, evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}_i}, \mathbf{B})$ and compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$. Set $\mathbf{A}_{\vec{V}_2} := [\mathbf{A} | \mathbf{B}_{\mathbf{v}_1} | \cdots | \mathbf{B}_{\mathbf{v}_k} | \mathbf{B}_{\mathbf{v}_{k+1}}]$ with $\vec{V}_2 := \vec{V}_1 \cup \{\mathbf{v}_{k+1}\}$. Note that, $\mathbf{A}_{\vec{V}_1} := [\mathbf{A} | \mathbf{B}_{\mathbf{v}_1} | \cdots | \mathbf{B}_{\mathbf{v}_k}]$. Compute trapdoor $\mathbf{A}^{-1}_{\vec{V}_2,\sigma_0}$ using the trapdoor $\mathbf{A}^{-1}_{\vec{V}_1,\sigma_0}$ (via Item 3 of Lemma 4) and output $\mathsf{sk}_{\vec{V}_2} := \mathbf{A}^{-1}_{\vec{V}_2,\sigma_0}$.
- $\mathsf{DMIPE.Enc}(\mathsf{pp}, \mu, \mathbf{x})$: On input public parameters $\mathsf{pp}$, a message vectors $\mu := (\mu_1, \cdots, \mu_m) \in \{0,1\}^m$ and an attribute vector $\mathbf{x} \in \mathbb{Z}_q^d$, do the following: Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{R} \xleftarrow{\$} \{-1, 0, 1\}^{m \times md}$ and $\mathbf{e}_{\mathsf{in}}, \mathbf{e}_{\mathsf{out}} \leftarrow \chi^m$. Compute $\mathbf{c}_{\mathsf{in}} := \mathbf{s}^\top \mathbf{A} + \mathbf{e}_{\mathsf{in}}^\top \in \mathbb{Z}_q^m$, $\mathbf{c}_{\mathsf{mid}} := \mathbf{s}^\top (\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top \mathbf{R} \in \mathbb{Z}_q^{md}$, $\mathbf{c}_{\mathsf{out}} := \mathbf{s}^\top \mathbf{U} + \mathbf{e}_{\mathsf{out}}^\top + \mu \cdot \lceil q/2 \rceil \in \mathbb{Z}_q^m$. Output ciphertext $\mathsf{ct}_{\mathbf{x}} := (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{mid}}, \mathbf{c}_{\mathsf{out}})$.
- $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathsf{ct}_{\mathbf{x}})$: On input public parameters $\mathsf{pp}$, secret key $\mathsf{sk}_{\vec{V}} := \mathbf{A}^{-1}_{\vec{V}}$ associated with $\vec{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_k)$ and a ciphertext $\mathsf{ct}_{\mathbf{x}} := (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{mid}}, c_{\mathsf{out}})$ associated with $\mathbf{x} \in \mathbb{Z}_q^d$, do the following: For each vector $\mathbf{v}_i$, evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow$

$\mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}_i}, \mathbf{B})$ and compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$. Set $\mathbf{A}_{\vec{V}} := [\mathbf{A}|\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$ and compute $\mathbf{W} \leftarrow \mathbf{A}^{-1}_{\vec{V},\sigma_0}(\mathbf{U})$, i.e., $\mathbf{A}_{\vec{V},\sigma_0}\mathbf{W} = \mathbf{U} \pmod q$. For $i \in [k]$, compute $\mathbf{c}_{\mathbf{v}_i} := \mathbf{c}_{\mathsf{mid}}\mathbf{H}_{\mathbf{v}_i}$, i.e, $\mathbf{c}_{\mathbf{v}_i} = \mathbf{s}^\top(\mathbf{B}_{\mathbf{v}_i} + \langle \mathbf{v}_i, \mathbf{x}\rangle \cdot \mathbf{G}) + \mathbf{e}^\top_{\mathsf{in}}\mathbf{R}\mathbf{H}_{\mathbf{v}_i}$. Compute $\mu' := (\mu'_1, \cdots, \mu'_m) \leftarrow \mathbf{c}_{\mathsf{out}} - [\mathbf{c}_{\mathsf{in}}|\mathbf{c}_{\mathbf{v}_1}|\cdots|\mathbf{c}_{\mathbf{v}_k}]\mathbf{W}$. For $i \in [m]$, output $\mu_i = 0$ if $|\mu'_i| < q/4$; output $\mu_i = 1$ otherwise.

### 5.1 Correctness, Parameters and Security Proofs

**Theorem 3 (Correctness).** *The given DMIPE is correct assuming the chosen parameters satisfy* $B + 12(mB + km^3B) \cdot \sigma_0 < q/4$.

*Proof.* See Appendix A. □

We choose the parameters as follows: (i) First, choose $\lambda$ to be a security parameter. (ii) For the hardness of $(n, 2m, q, \chi)$-DLWE (in Lemma 10): by [9, Lemma 3.2], we choose $n = n(\lambda)$, $\epsilon$, $q = q(n) \leq 2^n$, $m = \Theta(n \log q) = \mathsf{poly}(n)$, $\chi = \chi(n)$ such that $\chi$ is a $(B, \epsilon)$-bounded for some $B = B(n)$ such that, $q/B \geq 2^{n^\epsilon}$. Note that the "core-SVP hardness" methodology has been usually used in the literature for choosing practical parameters; see [5, Section 5.2.1]. (iii) $m > (n+1)\log q + \omega(\log n)$ (For Lemma 8; due to Lemma 3). (iv) $\sigma_0 = \omega(n \log q \log n)$ (for $\mathsf{TrapGen}$; due to Item 1 of Lemma 4).$\sigma \geq m^2 d \cdot \omega(\sqrt{\log m})$ (for $\mathsf{Hybrid}$ 3 to work; due to Lemma 9). (v) $B + 12(mB + km^3B) \cdot \sigma_0 < q/4$. (for Correctness; due to Theorem 3).

Now, we come up with the selective payload-hiding security of the proposed DMIPE.

**Theorem 4 (Selective Payload-hiding Security).** *Under the hardness of the* $(n, 2m, q, \chi)$-$\mathsf{DLWE}$ *assumption, the lattice-based* $\mathsf{DMIPE}$ *is selectively payload-hiding secure (under chosen plaintext attacks). Specifically, suppose that there is an adversary* $\mathcal{A}$ *that wins the* $\mathsf{DMIPE}^{\mathsf{sel,CPA}}_{\mathsf{payload},\mathcal{A}}$*, then one can use* $\mathcal{A}$ *to build a solver* $\mathcal{B}$ *that can solve the* $(n, 2m, q, \chi)$-$\mathsf{DLWE}$ *problem at least with the same advantage.*

*Proof.* We prove the theorem via a sequence of hybrids. Let $\mathcal{W}_i$ be the event $b' = b$ in $\mathsf{Hybrid}$ $i$. We want to prove that $|\Pr[\mathcal{W}_0]| \leq \mathsf{negl}(\lambda)$

$\mathsf{Hybrid}$ 0. This is the original game $\mathsf{DMIPE}^{\mathsf{sel,CPA}}_{\mathsf{payload},\mathcal{A}}$ stated in Figure 3. Suppose that the target attribute vector is $\mathbf{x}^*$ and the short matrix used in Step 1 of $\mathsf{DMIPE.Enc}$ for producing the challenge ciphertext (in the **Challenge** phase) is $\mathbf{R}^* \in \{-1, 0, 1\}^{m \times md}$.

$\mathsf{Hybrid}$ 1. This hybrid is similar to $\mathsf{Hybrid}$ 0 except that $\mathbf{R}^* \xleftarrow{\$} \{-1, 0, 1\}^{m \times md}$ is generated in the **Setup** phase instead in the **Challenge** phase.

$\mathsf{Hybrid}$ 2. This hybrid is similar to $\mathsf{Hybrid}$ 1 except the way the challenger sets public parameters $\mathsf{pp}$. Namely, $\mathsf{pp} := (\mathbf{A}, \mathbf{B}, \mathbf{U})$, where $\mathbf{B}$ is generated as $\mathbf{B} := \mathbf{A}\mathbf{R}^* + \mathbf{x}^* \otimes \mathbf{G} \in \mathbb{Z}_q^{n \times md}$, while $\mathbf{A}, \mathbf{U}$ are unchanged (i.e., $(\mathbf{A}, \mathbf{A}^{-1}_{\sigma_0}) \leftarrow$

$\mathsf{TrapGen}(n, m, q)$, $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}$). Note that, in this game the component $\mathbf{c}^*_{\mathsf{mid}}$ in the challenge ciphertext $\mathsf{ct}^*_{\mathbf{x}^*} = (\mathbf{c}^*_{\mathsf{in}}, \mathbf{c}^*_{\mathsf{mid}}, \mathbf{c}^*_{\mathsf{out}})$ can be rewritten as

$$\mathbf{c}^*_{\mathsf{mid}} := \mathbf{s}^\top(\mathbf{B} - \mathbf{x}^* \otimes \mathbf{G}) + \mathbf{e}^\top_{\mathsf{in}}\mathbf{R}^* = \mathbf{s}^\top(\mathbf{A}\mathbf{R}^*) + \mathbf{e}^\top_{\mathsf{in}}\mathbf{R}^* = \mathbf{c}^*_{\mathsf{in}}\mathbf{R}^*.$$

Hybrid 3. This hybrid is similar to Hybrid 2 except the way the challenger generates $\mathbf{A}$ in the public parameters $\mathsf{pp}$. Namely, $\mathbf{A}$ is sampled uniformly at random from $\mathbb{Z}_q^{n\times m}$ (and the challenger does not have $\mathbf{A}^{-1}_{\sigma_0}$). Instead, the challenger uses the trapdoor $\mathbf{G}^{-1}_{O(1)}$ for $\mathbf{G}$ as the master secret key. By this way, for any key query $\mathsf{KQ}(\vec{V})$, any key delegation query $\mathsf{KD}(\vec{V}, \mathbf{v})$, the challenger utilizes $\mathbf{G}^{-1}_{O(1)}$ to compute $\mathsf{sk}_{\vec{V}}$ with the help of Item 4 of Lemma 4. Specifically,

1. For each vector $\mathbf{v}_i$, evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}_i}, \mathbf{A}\mathbf{R}^*)$.
2. Compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i} = \mathbf{A}\mathbf{R}^*\mathbf{H}_{\mathbf{v}_i} + \langle \mathbf{v}_i, \mathbf{x}^*\rangle \cdot \mathbf{G}$.
3. Set $\mathbf{A}_{\vec{V}} := [\mathbf{A}|\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$, where $k = |\vec{V}|$.
4. Note that, if $\langle \mathbf{v}_i, \mathbf{x}^*\rangle = 0 \pmod q$ for all $\mathbf{v}_i \in \vec{V}$, then the challenger aborts the game. Otherwise, we will have at least one $\langle \mathbf{v}_{i_0}, \mathbf{x}^*\rangle \neq 0 \pmod q$, then the challenger can successfully
   (a) compute $[\mathbf{A}|\mathbf{A}\mathbf{R}^*\mathbf{H}_{\mathbf{v}_{i_0}} + \langle \mathbf{v}_{i_0}, \mathbf{x}^*\rangle \cdot \mathbf{G}]^{-1}_\sigma$ from $\mathbf{G}^{-1}_{O(1)}$ (Item 4 of Lemma 4),
   (b) compute $[\mathbf{A}_{\vec{V}}]^{-1}_\sigma$ from $[\mathbf{A}|\mathbf{A}\mathbf{R}^*\mathbf{H}_{\mathbf{v}_{i_0}} + \langle \mathbf{v}_{i_0}, \mathbf{x}^*\rangle \cdot \mathbf{G}]^{-1}_\sigma$ (Item 3 of Lemma 4), and finally assign $\mathsf{sk}_{\vec{V}} \leftarrow [\mathbf{A}_{\vec{V}}]^{-1}_\sigma$.

Hybrid 4. This hybrid is similar to Hybrid 3 except that for the challenge ciphertext $\mathbf{c}_{\mathsf{in}}$ and $\mathbf{c}_{\mathsf{out}}$ are both sampled uniformly at random from $\mathbb{Z}_q^m$, while $\mathbf{c}_{\mathsf{mid}}$ is unchanged. Obviously, $|\Pr[\mathcal{W}_4] - 1/2| = 0$.

We prove that $|\Pr[\mathcal{W}_0] - 1/2| \leq \mathsf{negl}(\lambda)$ through the following lemmas, which show the indistinguishability of the two consecutive hybrids above.

**Lemma 7.** *In the view of the adversary $\mathcal{A}$,* Hybrid 1 *and* Hybrid 0 *are perfectly the same; i.e.,* $\Pr[\mathcal{W}_1] = \Pr[\mathcal{W}_0]$.

**Lemma 8.** *In the view of the adversary $\mathcal{A}$,* Hybrid 2 *and* Hybrid 1 *are indistinguishable, i.e.,* $|\Pr[\mathcal{W}_1] - \Pr[\mathcal{W}_2]| = \mathsf{negl}(\lambda)$.

**Lemma 9.** *In the view of the adversary $\mathcal{A}$,* Hybrid 3 *and* Hybrid 2 *are indistinguishable, i.e.,* $|\Pr[\mathcal{W}_2] - \Pr[\mathcal{W}_3]| = \mathsf{negl}(\lambda)$.

**Lemma 10.** *In the view of the adversary $\mathcal{A}$,* Hybrid 4 *and* Hybrid 3 *are indistinguishable, i.e.,* $|\Pr[\mathcal{W}_3] - \Pr[\mathcal{W}_4]| = \mathsf{negl}(\lambda)$, *assuming the hardness of the* $(n, 2m, q, \chi)$-$\mathsf{DLWE}$ *problem.*

Proofs for Lemma 7-10 are included in Appendix A. Now, $|\Pr[\mathcal{W}_0] - 1/2| \leq |\Pr[\mathcal{W}_0] - \Pr[\mathcal{W}_1]| + |\Pr[\mathcal{W}_1] - \Pr[\mathcal{W}_2]| + |\Pr[\mathcal{W}_2] - \Pr[\mathcal{W}_3]| + |\Pr[\mathcal{W}_3] - \Pr[\mathcal{W}_4]| + |\Pr[\mathcal{W}_4] - 1/2| = \mathsf{negl}(\lambda)$. □

## 6 Constructing DMIPE from SE

One can construct DMIPE from SE. The key idea is that for predicate vectors $\vec{V}$, we utilize a transformation named $\mathsf{OVS}$, that maps $\vec{V}$ to the (unique) orthogonal complement of the subspace generated by all vectors in $\vec{V}$. That is, $\mathsf{OVS}(\vec{V}) := (\mathsf{span}(\mathbf{v}_1, \cdots, \mathbf{v}_k))^{\perp}$. Remind that all vectors in $\vec{V}$ are linearly independent. By doing that, the condition $\langle \mathbf{v}_i, \mathbf{x} \rangle = 0 \pmod q$ $\forall \mathbf{v}_i \in \vec{V}$ is equivalent to $\mathbf{x} \in \mathsf{OVS}(\vec{V})$. Furthermore, the transformation also guarantees that if $\vec{V}_1 \subseteq \vec{V}_2$ then $\mathsf{OVS}(\vec{V}_2) \sqsubseteq \mathsf{OVS}(\vec{V}_1)$.

The construction for DMIPE from SE is quite similar to the way for SE from DMIPE. We include it here for completeness. The correctness of DMIPE is straightforward from that of SE. The security of DMIPE follows from that of SE and can be done similarly as in the proof of Theorem 2. Then we omit it.

- $\mathsf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$: Run $(\mathsf{se.pp}, \mathsf{se.msk}) \leftarrow \mathsf{SE.Setup}(1^\lambda, \mathsf{sp})$ and then output $\mathsf{pp} := \mathsf{se.pp}$, $\mathsf{msk} := \mathsf{se.msk}$.
- $\mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{T}, \mathsf{msk}, \vec{V})$: Run $V \leftarrow \mathsf{OVS}(\vec{V})$, $\mathsf{se.sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$ and then output $\mathsf{sk}_{\vec{V}} := \mathsf{se.sk}_V$.
- $\mathsf{DMIPE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$: Run $\mathsf{se.ct}_{\mathbf{x}} \leftarrow \mathsf{SE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$ and output $\mathsf{ct}_{\mathbf{x}} := \mathsf{se.ct}_{\mathbf{x}}$.
- $\mathsf{DMIPE.Del}(\mathsf{pp}, \vec{V}_1, \mathsf{sk}_{\vec{V}_1}, \mathbf{v})$: Compute $V_1 \leftarrow \mathsf{OVS}(\vec{V}_1)$, $V_2 \leftarrow \mathsf{OVS}(\vec{V}_1 \cup \{\mathbf{v}\})$, and then $\mathsf{se.sk}_{V_2} \leftarrow \mathsf{SE.Del}(\mathsf{pp}, \mathsf{se.sk}_{V_1}, V_2)$. (Note that, $\mathsf{se.sk}_{V_1} = \mathsf{sk}_{\vec{V}_1}$.) Finally, output $\mathsf{sk}_{\vec{V}_2} := \mathsf{se.sk}_{V_2}$.
- $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$: Return the output of $\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$.

## 7 Allow-/Deny-list Encryption from Spatial Encryption

### 7.1 Framework of ADE

Let $\lambda$ be a security parameter, $d = d(\lambda)$ be the maximum number of negative tags per ciphertext, and $a = a(\lambda)$ be the the maximum number of positive tags in the ADE system. Further, we denote the space of plaintexts, the negative tag space and the positive tag space by $\mathcal{M} = \mathcal{M}(\lambda)$, $\mathcal{T}^{(-)} = \mathcal{T}^{(-)}(\lambda)$ and by $\mathcal{T}^{(+)} = \mathcal{T}^{(+)}(\lambda)$, respectively.

**Syntax.** ADE is a tuple of the following algorithms $\mathsf{ADE}$=($\mathsf{ADE.Gen}$, $\mathsf{ADE.Enc}$, $\mathsf{ADE.Npun}$, $\mathsf{ADE.Ppun}$, $\mathsf{ADE.Dec}$):

- $(\mathsf{pp}, \mathsf{sk}_\emptyset^\emptyset) \leftarrow \mathsf{ADE.Gen}(1^\lambda, 1^a, 1^d)$: On input (a security parameter $\lambda$ and a maximum number $a$ of positive tags per ciphertext and a maximum number $d$ of negative tags per ciphertext), the PPT algorithm outputs public parameters $\mathsf{pp}$ and a (not punctured) initial secret key $\mathsf{sk}_\emptyset^\emptyset$.
- $\mathsf{sk}_{DL'}^{AL'_1 \cup AL'_2} \leftarrow \mathsf{ADE.Ppun}(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'_1}, AL'_2, k)$: On input a tuple of (public parameters $\mathsf{pp}$; a previously punctured key $\mathsf{sk}_{DL'}^{AL'_1}$ w.r.t a set of positive tags $\emptyset \subseteq AL'_1 \subseteq \mathcal{T}^{(+)}$ and a set of negative tags $\emptyset \subseteq DL' \subseteq \mathcal{T}^{(-)}$; a set of positive tags $AL'_2 \in \mathcal{T}^{(+)} \setminus AL'_1$), the PPT algorithm returns a new punctured key $\mathsf{sk}_{DN'}^{A'L \cup \{\mathsf{pt}\}}$. Here, note that $k$ is only used in the $k$-tADE variant.

- $\mathsf{sk}^{AL'}_{DL'_1 \cup DL'_2} \leftarrow \mathsf{ADE.Npun}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'_1}, DL'_2)$: On input a tuple of (public parameters $\mathsf{pp}$; a previously punctured key $\mathsf{sk}^{AL'}_{DL'}$ w.r.t a set of positive tags $\emptyset \subseteq AL' \subseteq \mathcal{T}^{(+)}$ and a set of negative tags $\emptyset \subseteq DL'_1 \subseteq \mathcal{T}^{(-)}$; a set of negative tags $DL'_2 \in \mathcal{T}^{(-)} \setminus DL'_1$), the PPT algorithm returns a new punctured key $\mathsf{sk}^{AL'}_{DL'_1 \cup DL'_2}$.
- $\mathsf{ct}^{AL}_{DL} \leftarrow \mathsf{ADE.Enc}(\mathsf{pp}, \mu, AL, DL)$: On input a tuple of (public parameters $\mathsf{pp}$; a plaintext $\mu$; a set of positive tags $AL$; a set of negative tags $DL$), the PPT algorithm returns a ciphertext $\mathsf{ct}^{AL}_{DL}$.
- $\mu/\perp \leftarrow \mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL})$: : On input a tuple of (public parameters $\mathsf{pp}$; a secret key $\mathsf{sk}^{AL'}_{DL'}$ associated with $AL' \subseteq \mathcal{T}^{(+)}$ and $DL' \subseteq \mathcal{T}^{(-)}$; a ciphertext $\mathsf{ct}^{AL}_{DL}$ associated with $AL \subseteq \mathcal{T}^{(+)}$ and $DL \subseteq \mathcal{T}^{(-)}$), the DPT algorithm outputs either a plaintext $\mu$ if decryption succeeds or $\perp$ otherwise.

**Correctness and ADE Variants.** Consider all $\lambda, a, d \in \mathbb{N}$, $\mu \in \mathcal{M}$, $\emptyset \subset AL, AL' \subseteq \mathcal{T}^{(+)}$, $\emptyset \subset DL, DL' \subseteq \mathcal{T}^{(-)}$, $(\mathsf{pp}, \mathsf{sk}^{\emptyset}_{\emptyset}) \leftarrow \mathsf{ADE.Gen}(1^\lambda, 1^a, 1^d)$, $\mathsf{ct}^{AL}_{DL} \leftarrow \mathsf{ADE.Enc}\ (\mathsf{pp}, \mu, AL, DL)$, and any punctured key $\mathsf{sk}^{AL'}_{DL'}$ generated using any combination of $\mathsf{ADE.Npun}$, and $\mathsf{ADE.Ppun}$ on $AL', DL'$. We define the correctness and classify ADE variants at the same time. All variants require that the initial key is always able to successfully decrypt a ciphertext i.e., $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{\emptyset}_{\emptyset}, \mathsf{ct}^{AL}_{DL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. However, when punctured, the additional correctness requirement varies for each variant. Specifically,

1. **Standard ADE (sADE).** If $(AL = AL') \wedge (DL \cap DL' = \emptyset)$ then $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. Otherwise, $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}) = \mu] \leq \mathsf{negl}(\lambda)$.
2. **Inclusive ADE (iADE).** If $((AL' \subseteq AL) \wedge (DL \cap DL' = \emptyset))$ then $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. Otherwise, $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}) = \mu] \leq \mathsf{negl}(\lambda)$.
3. **$k$-threshold ADE ($k$-tADE).** If $((|AL \cap AL'| \geq k) \wedge (DL \cap DL' = \emptyset))$, then $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. Otherwise, $\Pr[\mathsf{ADE.Dec}\ (\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}) = \mu] \leq \mathsf{negl}(\lambda)$.

Note that, in iADE if the equality in $AL' \subseteq AL$ happens then we get sADE.

**Security Notions of ADE Variants.** Same as SE and DMIPE, one can define security following the PrE framework. However, we only focus on the notion of selective payload-hiding security for all ADE variants.

**Definition 3.** *ADE is selective payload-hiding secure if the advantage of the adversary playing in the* $\mathsf{ADE}^{\mathsf{sel,ATK}}_{\mathsf{payload},\mathcal{A}}$ *game (in Figure 4) is negligible or* $\mathsf{Adv}^{\mathsf{payload,ATK}}_{\mathsf{ADE},\mathcal{A},\mathsf{sel}} := |\Pr[b' = b] - 1/2| = \mathsf{negl}(\lambda)$.

### 7.2 Transforming sADE and iADE to SE

Let $\mathcal{T}^{(-)}, \mathcal{T}^{(+)} \subset \mathbb{Z}_q$ for $q$ prime. Suppose that we have at most $a$ positive tags and $d$ negative tags. i.e., $|\mathcal{T}^{(+)}| = a$ and $|\mathcal{T}^{(-)}| = d$ involved in the (s/i)ADE. For

GAME $\mathsf{ADE}^{\mathsf{sel},\mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}(\lambda, a, d)$:
(where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA1}, \mathsf{CCA2}\}$)
1. $(AL^*, DL^*) \leftarrow \mathcal{A}(1^\lambda, 1^a, 1^d)$; 2. $(\mathsf{pp}, \mathsf{sk}^\emptyset_\emptyset) \leftarrow \mathsf{ADE.Gen}(1^\lambda, 1^a, 1^d)$, $AL' \leftarrow \emptyset$, $DL' \leftarrow \emptyset$;
3. $(\mu^*_0, \mu^*_1) \leftarrow \mathcal{A}^{\mathsf{Punc}(\cdot,\cdot),\mathsf{DQ}(\cdot,\cdot)}(\mathsf{pp})$; 4. $b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}^{AL^*}_{DL^*} \leftarrow \mathsf{ADE.Enc}(\mathsf{pp}, \mu^*_b, AL^*, DL^*)$;
5. $b' \leftarrow \mathcal{A}^{\mathsf{Pun}(\cdot,\cdot),\mathsf{DQ}(\cdot)}(\mathsf{pp}, \mathsf{ct}^{AL^*}_{DL^*})$. // **NOTE:** Not allowed $\mathsf{DQ}(AL', DL', \mathsf{ct}^{AL^*}_{DL^*})$ with $(AL', DL') \in \mathsf{SUCC}(AL^*, DL^*)$;
6. If $b' = b$, return 1. Otherwise, return 0.

**Queried Oracles:**
- Puncturing Oracle $\mathsf{Pun}((AL', DL'))$ (It is only allowed if $(AL', DL') \notin \mathsf{SUCC}(AL^*, DL^*)$): Run $\mathsf{ADE.Ppun}$ and $\mathsf{ADE.Npun}$ in any order using $\mathsf{sk}^\emptyset_\emptyset$ to output $\mathsf{sk}^{AL'}_{DL'}$.
- Decryption Oracle $\mathsf{DQ}(AL', DL', \mathsf{ct}^{AL}_{DL})$ (allowed only if $\mathsf{ATK}$=$\mathsf{CCA}$): Run $\mathsf{ADE.Ppun}$ and $\mathsf{ADE.Npun}$ in any order using $\mathsf{sk}^\emptyset_\emptyset$ to get $\mathsf{sk}^{AL'}_{DL'}$. Finally, return the output of $\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL}, )$..

**Define $\mathsf{SUCC}(AL^*, DL^*)$ for ADE Variants:**
- **sADE:** $\mathsf{SUCC}(AL^*, DL^*) := \{(AL', DL') : ((AL' = AL^*) \wedge (DL' \cap DL^* = \emptyset))\}$ .
- $k$**-tADE**: $\mathsf{SUCC}(AL^*, DL^*) := \{(AL', DL') : ((|AL' \cap AL^*| \geq k) \wedge (DL' \cap DL^* = \emptyset))\}$ .
- **iADE:** $\mathsf{SUCC}(AL^*, DL^*) := \{(A'L, DL') : ((AL' \subseteq AL^*) \wedge (DL' \cap DL^* = \emptyset))\}$ .

Fig. 4: Selective security for the ADE variants

$(AL'_1, DL'_1), (AL'_2, DL'_2) \in \mathcal{T}^{(+)} \times \mathcal{T}^{(-)}$, we say $(AL'_1, DL'_1) \subseteq (AL'_2, DL'_2)$ if and only if $(AL'_1 \subseteq AL'_2) \wedge (DL'_1 \subseteq DL'_2)$. Now, for any pair $(AL', DL') \subseteq \mathcal{T}^{(+)} \times \mathcal{T}^{(-)}$ punctured on decryption keys, we will try to encode it as a (possibly affine) subspace $V$ compatible with the SE syntax. On the other hand, for any pair $(AL, DL)$ of positive/negative ciphertext tags, we will try to encode it as a vector $\mathbf{v}$ such that $\mathbf{v} \in V$ iff $(AL' \subseteq AL) \wedge (DL' \cap DL = \emptyset)$. We need the following encodings $\mathsf{EncodeInKey}$ and $\mathsf{EncodeInCipher}$ to do that:

- $W_{\mathsf{key}} \leftarrow \mathsf{EncodeInKey}(AL', DL')$. Do the following: Associate the allow list $AL' = \{p_1, \cdots, p_k\}$ with a space beginning with $(p_1, \cdots, p_k)^\top$, namely $W_{AL'} := \{(p_1, \cdots, p_k, x_{k+1}, \cdots, x_a)^\top : x_i \in \mathbb{Z}_q\} \subseteq \mathbb{Z}^a_q$. Obviously, it is easy to see that if $AL'_1 \subseteq AL'_2$ then $W_{AL'_2} \sqsubseteq W_{AL'_1}$. For the deny list $DL'$, compute its complement $DL'' := \mathcal{T}^{(-)} \setminus DL'$ then associate $DL'$ with $W_{DL'} := \mathsf{span}\{\mathbf{v}_x : x \in DL''\}$, where $\mathbf{v}_x := (1, x, x^2, \cdots, x^{2d-1})$ is a Vandermonde vector. Since adding more tags into $DL'$ is equivalent to removing tags from $DL''$, then given $DL'_1 \subseteq DL'_2$ we have $W_{DL''_2} \sqsubseteq W_{DL''_1}$. Output the subspace $W_{\mathsf{key}}$ which is the direct product of $W_{AL'}$ and $W_{DL'}$: $W_{\mathsf{key}} := W_{AL'} \times W_{DL'}$.
- $\mathbf{x}_{\mathsf{ct}} \leftarrow \mathsf{EncodeInCipher}(AL, DL)$. Do the following steps: For $AL = \{p_1, \cdots, p_k\}$, associate $AL$ with vector $\mathbf{x}_{AL} := (p_1, \cdots, p_k, 0, \cdots, 0) \in \mathbb{Z}^a_q$. Clearly, if $AL' \subseteq AL$ then $\mathbf{x}_{AL} \in W_{AL'}$. For a list $DL$, encode it as $\mathbf{x}_{DL} := \sum_{x \in DL} \mathbf{v}_x \in \mathbb{Z}^{2d}_q$, where $\mathbf{v}_x := (1, x, x^2, \cdots, x^{2d-1})$. We claim that $\mathbf{x}_{DL} \notin W_{DL'}$ for any $DL \cap DL' \neq \emptyset$ (i.e., $DL \nsubseteq DL''$ ). Output vector $\mathbf{x}_{\mathsf{ct}} := (\mathbf{x}_{AL}, \mathbf{x}_{DL}) \in \mathbb{Z}^{a+2d}_q$.

We can see that $\mathbf{x}_{\mathsf{ct}} \in W_{\mathsf{key}}$ iff $(\mathbf{x}_{AL}, \mathbf{x}_{DL}) \in W_{AL'} \times W_{DL'}$, which is equivalent to $(AL' \subseteq AL) \wedge (DL' \cap DL = \emptyset)$. Therefore, the correctness of (s/i)ADE can be straightforwardly obtained from the correctness of SE.

One can easily see that puncturings of (s/i)ADE can be done through delegation of SE.

## 8 Conclusions and Future Works

We revisit SE towards an efficient lattice-based SE. Along the way, we introduce the new concept of DMIPE. We show that DMIPE is sufficient for building an efficient lattice-based SE. The lattice-based SE is more efficient than some previous lattice-based ones, which follow the generic SE construction from the HIPE. Moreover, DMIPE and SE are equivalent in the sense that there are "security notions-preserving" conversions between them.

Although our lattice-based DMIPE is proven to be selectively payload-hiding secure in the standard model, the construction can enjoy selectively weak attribute-hiding security. A possible technical idea might be from Agrawal et al. [3]. However, we leave this for future work. Furthermore, an adaptively secure DMIPE construction in the lattice setting is a worthwhile pursuit in the future. Recall that such construction for IPE has been done by [24]. Additionally, an attribute-hiding secure DMIPE construction over lattices should also be interesting for further research. Also, as mentioned before, we leave open the encodings for transforming $k$-tADE to SE. We think that the idea of *threshold gates* in Hamburg's thesis [23, Page 51] can help. However, the Doubly Spatial Encryption (DSE) or another SE variant rather than the original SE (as defined in this paper) might be needed.

**Acknowledgment.** This work is partially supported by the Australian Research Council Linkage Project LP190100984. Huy Quoc Le has been sponsored by a CSIRO Data61 PhD Scholarship and CSIRO Data61 Top-up Scholarship. Josef Pieprzyk has been supported by the Polish National Science Center (NCN) grant 2018/31/B/ST6/03003.

## References

1. Abdalla, M., De Caro, A., Mochetti, K.: Lattice-based hierarchical inner product encryption. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7533 LNCS**, 121–138 (2012). https://doi.org/10.1007/978-3-642-33481-8_7
2. Agrawal, S., Boneh, D., Boyen, X.: Efficient Lattice (H)IBE in the Standard Model. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. vol. 6110 LNCS, pp. 553–572. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_28
3. Agrawal, S., Freeman, D.M., Vaikuntanathan, V.: Functional encryption for inner product predicates from learning with errors. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7073 LNCS**, 21–40 (2011). https://doi.org/10.1007/978-3-642-25385-0_2

4. Ajtai, M.: Generating Hard Instances of Lattice Problems (Extended Abstract). In: Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing. pp. 99–108. STOC '96, ACM, New York, NY, USA (1996). https://doi.org/10.1145/237814.237838
5. Alkim, E., Bos, J.W., Ducas, L., others: Frodo{KEM}: Learning with Errors Key Encapsulation Algorithm Specifications And Supporting Documentation, version 30 September, 2020. Tech. rep. (2020)
6. Boneh, D., Gentry, C., Gorbunov, S., Halevi, S., Nikolaenko, V., Segev, G., Vaikuntanathan, V., Vinayagamurthy, D.: Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 8441 LNCS, pp. 533–556 (2014). https://doi.org/10.1007/978-3-642-55220-5_30
7. Boneh, D., Hamburg, M.: Generalized identity based and broadcast encryption schemes. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **5350 LNCS**, 455–470 (2008). https://doi.org/10.1007/978-3-540-89255-7_28
8. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **4392 LNCS**, 535–554 (2007). https://doi.org/10.1007/978-3-540-70936-7_29
9. Brakerski, Z., Vaikuntanathan, V.: Circuit-ABE from LWE: Unbounded Attributes and Semi-adaptive Security. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016. vol. 9816, pp. 363–384. Springer, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_13
10. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010. EUROCRYPT 2010. Lecture Notes in Computer Science. vol. 6110, pp. 601–639. Springer-Verlag, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_27
11. Chen, C., Zhang, Z., Feng, D.: Fully secure doubly-spatial encryption under simple assumptions. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **7496 LNCS**, 253–263 (2012). https://doi.org/10.1007/978-3-642-33272-2_16
12. Chen, J., Lim, H., Ling, S., Wang, H.: The relation and transformation between hierarchical inner product encryption and spatial encryption. Designs, Codes, and Cryptography **71**(2), 347–364 (2014). https://doi.org/10.1007/s10623-012-9742-y
13. Chen, J., Wee, H.: Doubly spatial encryption from DBDH. Theoretical Computer Science **543**(C), 79–89 (2014). https://doi.org/10.1016/j.tcs.2014.06.003
14. Cohen, H.: A Course in Computational Algebraic Number Theory. No. Graduate texts in mathematics, 138, Springer, Berlin (1996)
15. Derler, D., Krenn, S., Lorünser, T., Ramacher, S., Slamanig, D., Striecks, C.: Revisiting Proxy Re-encryption: Forward Secrecy, Improved Security, and Applications. In: Abdalla, M., Dahab, R. (eds.) Public-Key Cryptography – PKC 2018. pp. 219–250. Springer International Publishing, Cham (2018). https://doi.org/10.1007/978-3-319-76578-5_8
16. Derler, D., Ramacher, S., Slamanig, D., Striecks, C.: Fine-Grained Forward Secrecy : Allow-List / Deny-List Encryption and Applications. In: Financial Cryptography and Data Security 2021. pp. 1–22 (2021)

17. Dutta, P., Susilo, W., Duong, D.H., Roy, P.S.: Puncturable Identity-Based Encryption from Lattices. In: Baek, J., Ruj, S. (eds.) Australasian Conference on Information Security and Privacy. pp. 571–589. Springer International Publishing, Cham (2021). https://doi.org/10.1007/978-3-030-90567-5_29
18. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions. In: Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing. pp. 197–206. STOC '08, ACM, New York, NY, USA (2008). https://doi.org/10.1145/1374376.1374407
19. Gentry, C., Sahai, A., Waters, B.: Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013. pp. 75–92. Springer Berlin Heidelberg, Berlin, Heidelberg (2013)
20. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) Advances in Cryptology — ASIACRYPT 2002. pp. 548–566. Springer Berlin Heidelberg, Berlin, Heidelberg (2002). https://doi.org/10.1007/3-540-36178-25C_34
21. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the ACM Conference on Computer and Communications Security. pp. 89–98 (2006). https://doi.org/10.1145/1180405.1180418
22. Green, M.D., Miers, I.: Forward Secure Asynchronous Messaging from Puncturable Encryption. In: 2015 IEEE Symposium on Security and Privacy. pp. 305–320 (5 2015). https://doi.org/10.1109/SP.2015.26
23. Hamburg, M.: Spatial Encryption. PhD. Thesis (July) (2011)
24. Katsumata, S., Nishimaki, R., Yamada, S., Yamakawa, T.: Adaptively Secure Inner Product Encryption from LWE. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **12493 LNCS**, 375–404 (2020). https://doi.org/10.1007/978-3-030-64840-4_13
25. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **4965 LNCS**(2006), 146–162 (2008). https://doi.org/10.1007/978-3-540-78967-3_9
26. Lyubashevsky, V.: Lattice Signatures without Trapdoors. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 738–755. Springer Berlin Heidelberg, Berlin, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
27. Micciancio, D., Peikert, C.: Trapdoors for Lattices: Simpler, Tighter, Faster, Smaller. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. pp. 700–718. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
28. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 5912 LNCS, pp. 214–231 (2009). https://doi.org/10.1007/978-3-642-10366-7_13
29. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56**(6), 84–93 (9 2009). https://doi.org/10.1145/1568318.1568324
30. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakley, G.R., Chaum, D. (eds.) Advances in Cryptology. vol. 196 LNCS, pp. 47–53. Springer Berlin Heidelberg, Berlin, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_5

31. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (11 2002). https://doi.org/10.1109/sfcs.1994.365700
32. Susilo, W., Duong, D.H., Le, H.Q., Pieprzyk, J.: Puncturable encryption: A generic construction from delegatable fully key-homomorphic encryption. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **12309 LNCS**, 107–127 (2020). https://doi.org/10.1007/978-3-030-59013-0_6
33. Tsabary, R.: Fully Secure Attribute-Based Encryption for t-CNF from LWE, vol. 11692 LNCS. Springer International Publishing (2019). https://doi.org/10.1007/978-3-030-26948-7_3
34. Xagawa, K.: Improved ( Hierarchical ) Inner-Product Encryption from Lattices. Full version of the paper appeared at PKC'13 pp. 235–252 (2015)
35. Zhou, M., Cao, Z.: Spatial encryption under simpler assumption. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **5848 LNCS**, 19–31 (2009). https://doi.org/10.1007/978-3-642-04642-1_4

## A Proofs

*Proof of Theorem 2.* The adversary $\mathcal{A}$ will take the role of the SE challenger playing with $\mathcal{S}$. Furthermore, the winning strategy of $\mathcal{A}$ is to simulate the environment of the same security game for $\Pi_{\mathsf{SE}}$ for the SE adversary $\mathcal{S}$ to join. The reduction is as follows.

**Setup.** After getting the public parameters $\mathsf{pp}$ from the DMIPE challenger $\mathcal{C}$, $\mathcal{A}$ hands $\mathsf{pp}$ to $\mathcal{S}$.

**Query 1.** For any query receiving from $\mathcal{S}$, $\mathcal{A}$ first uses the algorithm $\mathsf{OCB}$ to convert the queries into the forms compatible with DMIPE. $\mathcal{A}$ then forwards those to $\mathcal{C}$. $\mathcal{A}$ responds $\mathcal{S}$ with what $\mathcal{C}$ sent back to $\mathcal{A}$ .

**Challenge.** The adversary $\mathcal{S}$ now submits its challenge (plaintexts and/or attribute vectors). The adversary $\mathcal{A}$ then forwards these to $\mathcal{C}$. Finally, $\mathcal{A}$ forwards to $\mathcal{S}$ what $\mathcal{C}$ has returned.

**Query 2.** Same as **Query 1** with the restriction mentioned in both DMIPE and SE games.

**Output.** Finally, $\mathcal{A}$ will output $b'$ which $\mathcal{S}$ has guessed.

**Analysis.** Obviously, the SE game environment that $\mathcal{A}$ simulated for $\mathcal{S}$ is perfect in the view of $\mathcal{S}$. Therefore, the winning advantage of $\mathcal{A}$ is at least as same as that of $\mathcal{S}$. □

*Proof of Lemma 5.* We give such a construction of $\mathbf{H}$, which in turn proves the existence of the algorithm $\mathsf{EvalF}^{\mathsf{IP}}$. Assume that $\mathbf{v} = (v_1, \cdots, v_d) \in \mathbb{Z}_q^d$. For $i \in [d]$, let $\mathbf{H}_i := \mathbf{G}^{-1}(v_i\mathbf{G}) \in \{0,1\}^{m\times m}$. Note that, $\mathbf{G}\mathbf{H}_i = v_i\mathbf{G}$. Now just let $\mathbf{H} := \begin{bmatrix}\mathbf{H}_1\\ \vdots \\ \mathbf{H}_d\end{bmatrix} \in \{0,1\}^{md\times m}$ then $(\mathbf{x}\otimes\mathbf{G})\mathbf{H} = \sum_{i=1}^{d} x_i\mathbf{G}(\mathbf{G}^{-1}(v_i\mathbf{G})) = \sum_{i=1}^{d} x_i v_i\mathbf{G} = \langle\mathbf{v},\mathbf{x}\rangle\cdot\mathbf{G}$. Therefore, $[\mathbf{B}\pm\mathbf{x}\otimes\mathbf{G}]\mathbf{H} = \mathbf{B}\mathbf{H}\pm\langle\mathbf{v},\mathbf{x}\rangle\cdot\mathbf{G} \pmod q$. Furthermore, $\|\mathbf{H}\|_{\max} \le 1$ as $\mathbf{H}\in\{0,1\}^{md\times m}$. □

*Proof of Theorem 3.* We have $\mathbf{c}_{\mathbf{v}_i} = \mathbf{s}^\top(\mathbf{B}_{\mathbf{v}_i} - \langle \mathbf{v}_i, \mathbf{x}\rangle \cdot \mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_i} = \mathbf{s}^\top \mathbf{B}_{\mathbf{v}_i} + \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_i}$ if and only if $\langle \mathbf{v}_i, \mathbf{x}\rangle = 0$. Therefore, if $\langle \mathbf{v}_i, \mathbf{x}\rangle = 0$ for all $\mathbf{v}_i \in \vec{V}$, then

$$\begin{aligned}\mu' :&= \mathbf{c}_{\mathsf{out}} - [\mathbf{c}_{\mathsf{in}} | \mathbf{c}_{\mathbf{v}_1} | \cdots | \mathbf{c}_{\mathbf{v}_k}]\mathbf{W} \\ &= \mu \cdot \lceil q/2 \rceil + \mathbf{e}_{\mathsf{out}} + [\mathbf{e}_{\mathsf{in}}^\top | \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_1} | \cdots | \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_k}]\mathbf{W}.\end{aligned}$$

Therefore,

$$\begin{aligned}&\|\mathbf{e}_{\mathsf{out}} + [\mathbf{e}_{\mathsf{in}}^\top | \mathbf{e}_{\mathsf{in}}^\top \mathbf{R} t \mathbf{H}_{\mathbf{v}_1} | \cdots | \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_k}]\mathbf{W}\|_{\max} \\ &\le \|\mathbf{e}_{\mathsf{out}}\|_{\max} + [\mathbf{e}_{\mathsf{in}}^\top | \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_1} | \cdots | \mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_k}]\mathbf{W}\|_{\max} \\ &\le \|\mathbf{e}_{\mathsf{out}}\|_{\max} + (m\|\mathbf{e}_{\mathsf{in}}^\top\|_{\max} + km \max_{i\in[k]} \|\mathbf{e}_{\mathsf{in}}^\top \mathbf{R}\mathbf{H}_{\mathbf{v}_i}\|\|_{\max}) \cdot \|\mathbf{W}\|_{\max} \\ &\le \|\mathbf{e}_{\mathsf{out}}\|_{\max} + (m\|\mathbf{e}_{\mathsf{in}}^\top\|_{\max} + km^3 \|\mathbf{e}_{\mathsf{in}}^\top\|_{\max} \cdot \|\mathbf{R}\|_{\max} \cdot \max_{i\in[k]} \|\mathbf{H}_{\mathbf{v}_i}\|_{\max}) \cdot \|\mathbf{W}\|_{\max} \\ &\le B + 12(mB + km^3 B) \cdot \sigma_0.\end{aligned}$$

Here, the second and the third inequality are due to Lemma 1. The last inequality is due to $\|\mathbf{e}_{\mathsf{in}}^\top\|_{\max} \le B$ (as $\chi$ is $B$-bounded), $\|\mathbf{R}\|_{\max} \le 1$, $\|\mathbf{H}_{\mathbf{v}_i}\|_{\max} \le 1$ (Lemma 5) and $\|\mathbf{W}\|_{\max} \le 12\sigma_0$ (by Item 5 of Lemma 4). By choosing parameters such that $B + 12(mB + km^3 B) \cdot \sigma_0 \le q/4$, the theorem follows. □

*Proof of Lemma 7.* The lemma follows from the fact that sampling $\mathbf{R}^*$ is independent of the view of $\mathcal{A}$. Hence, the challenger can sample $\mathbf{R}^*$ at any time before returning the challenge ciphertext without making the adversary notice. □

*Proof of Lemma 8.* This is simply due to the leftover hash lemma (Lemma 3). □

*Proof of Lemma 9.* This is simply due to (i) the pseudo-randomness of TrapGen (see Item 1 of Lemma 4 ) and (ii) the distribution of secret keys generated using the trapdoor $\mathbf{G}_{O(1)}^{-1}$ of $\mathbf{G}$ are the same as that generated using the trapdoor $\mathbf{A}_{\sigma_0}^{-1}$. However, we have to care about choosing the Gaussian parameter $\sigma$ in Step 4 of Hybrid 3. Namely, we should choose

$$\begin{aligned}\sigma &= m \cdot \|\mathbf{R}^*\mathbf{H}_{\mathbf{v}_{i_0}}\|_{\max} \cdot \omega(\sqrt{\log m}) \le m^2 d \cdot \|\mathbf{R}^*\|_{\max} \cdot \|\mathbf{H}_{\mathbf{v}_{i_0}}\|_{\max} \cdot \omega(\sqrt{\log m}) \\ &\le m^2 d \cdot \omega(\sqrt{\log m}).\end{aligned}$$

□

*Proof of Lemma 10.* Suppose that $\mathcal{A}$ can distinguish Hybrid 4 from Hybrid 3 with a non-negligible advantage. From $\mathcal{A}$, we construct a DLWE solver $\mathcal{B}$ as follows:

**DLWE Instance**. The DLWE solver $\mathcal{B}$ is required to solve an $(n, 2m, q, \chi)$-DLWE instance $(\mathbf{F}, \mathbf{c})$, with $\mathbf{F} \xleftarrow{\$} \mathbb{Z}_q^{n\times 2m}$, and a vector $\mathbf{c} \in \mathbb{Z}_q^{2m}$ is either (i) random or (ii) LWE samples, i.e., $\mathbf{c}^\top = \mathbf{s}^\top\mathbf{F} + \mathbf{e}^\top$, for some random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \leftarrow \chi^{2m}$.

**Initialize.** Now $\mathcal{B}$ calls $\mathcal{A}$ to get the target attribute vector $\mathbf{x}^* \in \mathbb{Z}_q^d$ upon which $\mathcal{A}$ wants to challenge.

**Setup.** $\mathcal{B}$ now simulates the environment for $\mathcal{A}$ by parsing $(\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{out}}) \leftarrow \mathbf{c}$, with $\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{out}} \in \mathbb{Z}_q^m$, $(\mathbf{e}_{\mathsf{in}}, \mathbf{e}_{\mathsf{out}}) \leftarrow \mathbf{e}$, where $\mathbf{e}_{\mathsf{in}}, \mathbf{e}_{\mathsf{out}} \leftarrow \chi^m$, and $(\mathbf{A}, \mathbf{U}) \leftarrow \mathbf{F}$, with

$\mathbf{A}, \mathbf{U} \in \mathbb{Z}_q^{n \times m}$. $\mathcal{B}$ generates the public parameters $\mathsf{pp} = (\mathbf{A}, \mathbf{B}, \mathbf{U})$ and master secret key as in Hybrid 3 by sampling $\mathbf{R}^* \xleftarrow{\$} \{-1, 0, 1\}^{m \times md}$ and then setting $\mathbf{B} := \mathbf{A}\mathbf{R}^* + \mathbf{x}^* \otimes \mathbf{G}$. After that, $\mathcal{B}$ sends $\mathsf{pp}$ to $\mathcal{A}$.

**Query.** For all sorts of queries that $\mathcal{A}$ makes, $\mathcal{B}$ replies similarly to Hybrid 3.

**Challenge.** At this point, $\mathcal{A}$ challenges by submitting two messages $\mu_0^*$ and $\mu_1^*$. In turn, $\mathcal{B}$ chooses a bit $b \xleftarrow{\$} \{0, 1\}$, then computes the challenge ciphertext $\mathsf{ct}_{\mathbf{x}^*} = (\mathbf{c}_{\mathsf{in}}^*, \mathbf{c}_{\mathsf{mid}}^*, \mathbf{c}_{\mathsf{out}}^*)$ by setting $\mathbf{c}_{\mathsf{in}}^* = \mathbf{c}_{\mathsf{in}}$, $\mathbf{c}_{\mathsf{mid}}^{*\top} \leftarrow \mathbf{c}_{\mathsf{in}}^{*\top}\mathbf{R}^*$ and $\mathbf{c}_{\mathsf{out}}^* \leftarrow \mathbf{c}_{\mathsf{out}} + \mu_b^* \lceil \frac{q}{2} \rceil$.

- If $\mathbf{c}$ is LWE samples, then $\mathbf{c}_{\mathsf{in}}^\top = \mathbf{s}^\top\mathbf{A} + \mathbf{e}_{\mathsf{in}}^\top$, $\mathbf{c}_{\mathsf{out}}^\top = \mathbf{s}^\top\mathbf{U} + \mathbf{e}_{\mathsf{out}}^\top$. Hence, $\mathbf{c}_{\mathsf{mid}}^{*\top} = \mathbf{c}_{\mathsf{in}}^{*\top}\mathbf{R}^* = \mathbf{s}^\top\mathbf{A}\mathbf{R}^* + \mathbf{e}_{\mathsf{in}}^\top\mathbf{R}^* = \mathbf{s}^\top(\mathbf{B} - \mathbf{x}^* \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top\mathbf{R}^*$, which is exactly the ones computed in Hybrid 3.
- If $\mathbf{c}$ is random then so are $\mathbf{c}_{\mathsf{in}}^*$, $\mathbf{c}_{\mathsf{out}}^*$. Hence, $\mathsf{ct}_{\mathbf{x}^*}^*$ is exactly computed as in Hybrid 4.

**Output.** $\mathcal{B}$ takes the $\mathcal{A}$'s output as its decision for the DLWE problem. □