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 the 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, Forward-secure cryptosystems.

In this paper, we revisit SE toward a more compact SE in the lattice setting. In doing that, we introduce a novel primitive called Delegatable Multiple Inner Product Encryption (DMIPE), which 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. The DMIPE design in turn implies a more compact lattice-based SE in terms of sizes, in comparison with SEs converted from HIPE (e.g., Xagawa's HIPE at PKC 2013) using the framework by Chen at al. (Designs, Codes, and Cryptography, 2014). Furthermore, we show that SE can also be used to implement the Allow-/Deny-list encryption, which subsumes, e.g., puncturable encryption (Green and Miers at IEEE S&P 2015) among others.

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

The notion of predicate encryption (PrE) introduced by Katz, Sahai and Waters [31] generalizes identity-based encryption (IBE) [39], attribute-based encryption

(ABE) [27] and hidden vector encryption (HVE) [12]. 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 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) [10], for instance.

Spatial Encryption. SE was introduced by Boneh and Hamburg in their paper [10] at Asiacrypt 2008 and then it was systematically investigated in the Hamburg's thesis [29]. SE is a subclass of PrE, in which predicates and attributes are relative to affine/vector objects. 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 to a linear SE, then it is sufficient to talk about linear SEs throughout this work.

In SE, encryption is done on an input pair of (plaintext, attribute), where the latter is a vector or an affine point (called *attribute vector* from now on). A decryption key is associated with a vector space or an affine space (called *predicate space* hereafter). The decryption key allows to recover the plaintext if and only if the attribute vector belongs to the predicate space. Given a space V and its subspace W, it is required that one can delegate a decryption key sk_W for W from a decryption key sk_V for V.

There exists also 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 and only if the intersection of the attribute and policy spaces is not empty. Because of important applications of SE (and DSE) for 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 [10], [29], [45], [17], [18], [19].

1.1 Our Motivations

Our work is inspired by a wide range of possible applications of SE and DSE as argued in [10] and [29]. However, the main driver behind our work is an attempt to remove shortcomings of a generic SE construction via [18]. Furthermore, as lattice-based cryptosystems are resistant against 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], [44].

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 [10], [29]). This is done by converting e.g., identities, time periods into vectors/spaces compatible with SE. For more details, the reader is referred to [10], [29]. Our further discussion is driven by the following questions.

Q1: "*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 puncturable encryption (PE) [28], [41], DFPE [22] and their gen-

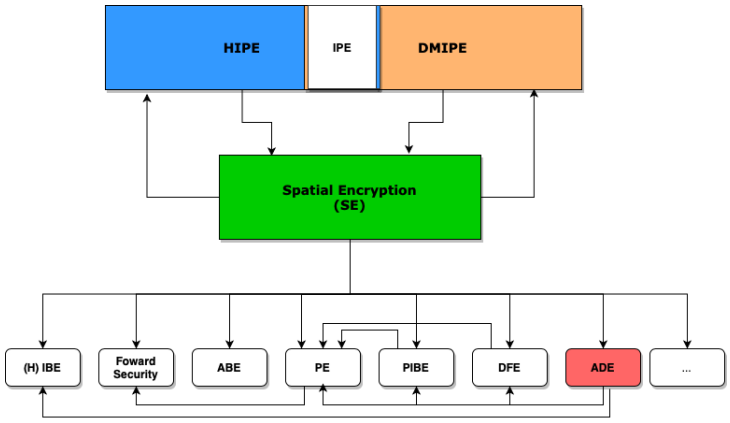

Fig. 1: The relation of SE and other primitives. Here, the implying relation A $\rightarrow$ B means that from A one can construct B.

eralization, the allow/deny-list encryption (ADE). ADE has also been mentioned by Derler et al. in their work [22] and can also subsume other "predecessors" such as IBE [39], HIBE [25], fs-PIBE [43], FuPE [21].

Roughly saying, ADE is an encryption framework endowed with a delegation mechanism, that we call *puncturing*, on *tags*. Tags are classified into two lists, *allow* and *deny*, which consists of *positive* and *negative* tags, respectively. A positive tag allows decryption while a negative one does not. These tags, with a varying quantity depending on case by case, are embedded into ciphertexts as well as decryption keys with different rules. E.g., in PE, only negative tags are punctured; while in DFPE, multiple positive tags and at most one positive tag can be punctured, in any order. (We refer to [22, Table 1] for a summary and comparison of ADE and its predecessors.) Generally, the main challenge in designing of ADE is to ensure that *positive puncturing* and *negative puncturing* on multiple tags can be done *in any order*. Note that so far there is neither formal definition nor security notions for ADE. We fill this gap and provide a formal definition of ADE (and its versions) as well as their security notions.

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) [10, 29], decisional bilinear Diffie-Hellman (DBDH) [17,19,45], and decisional linear (DLIN) [17]. Some of them are provably secure in the generic group model (GGM) [10], [29], while others – in the standard model (SDM) [45], [17], [19] .

Recall that lattice-based cryptosystems are believed to enjoy post-quantum (PQ) security. In contrast, cryptosystems whose security is based on intractability of factorisation or discrete logarithm are breakable by quantum adversaries [40]. With a 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.

Q2: "*Is it possible to design SE in the lattice setting?*"

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

We found out that an answer to the question has already existed in the literature. In fact, one can get a lattice-based SE using a generic construction from the hierarchical inner product encryption (HIPE) given by Chen et al. in [18]. The generic construction deploys two lattice-based SE schemes from [1] and [44]. Security of both schemes is based on intractability of the learning with errors (LWE) problem. Unfortunately, a such (lattice-based) SE construction is not free from few weaknesses.

Table 1: (D)SE constructions

Literature	**From**	**Assumption**	**Security model**	**Selective** (S) /**Adaptive** (A)	**PQ security**
Boneh, Hamburg [10]		BDDHE	GGM	S & A	✗
Hamburg [29]		BDDHE	GGM	S	✗
Chen, Wee [19]		DBDH	SDM	S	✗
Chen, Zhang, Feng [17]		DLIN, DBDH	SDM	A	✗
Zhou, Cao [45]		DBDH	SDM	S	✗
Abdalla, Caro, Mochetti [1]	HIPE via Chen [18]	LWE	SDM	S	✓
Xagawa [44]	HIPE via Chen [18]	LWE	SDM	S	✓
Our work	DMIPE	LWE	SDM	S	✓

Shortcomings of Constructing SE from HIPE. The notion of inner product encryption (IPE) is introduced by Katz, Sahai, and Waters [31]. IPE is a also subclass of PrE and can be seen as a generalization of IBE, in which each ciphertext and decryption key involve a vector defined over a finite field. A decryption key can recover a plaintext from a ciphertext if the inner product of two vectors is equal to zero. Hierarchical inner product encryption (HIPE) is then introduced as an extension of IPE by Okamoto and Takashima [35]. In HIPE, a list of attribute vectors is embedded into a ciphertext and another list of predicate vectors – in a decryption key. To decrypt a plaintext, a complex collection of requirements has to be satisfied.

We now briefly introduce HIPE to the reader. Let $\mathbb{F}$ be a field and let $\Delta(\delta) := (\delta; \ell_1, \cdots, \ell_\delta)$ be a tuple of positive integers, which is called a *hierarchical format* of depth δ. For $k \leq \delta$, we consider $\Gamma_{|k} := \Gamma_1 \times \cdots \times \Gamma_k$, where $\Gamma_i := \mathbb{F}^{\ell_i}$. For any $\vec{V} = (\mathbf{v}_1, \cdots, \mathbf{v}_k) \in \Gamma_{|k}$, 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, given $\Delta(\delta)$-HIPE. One encrypts a plaintext together with a list of attribute vectors $\vec{X}$ and produces a decryption key together with a list of predicate vectors $\vec{V}$. The criteria for successful decryption is that the hierarchical predicate $f_{\vec{V}}(\vec{X})$ defined above is equal to 1.

Chen et al. [18] have investigated a 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 9 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's why in a 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 a 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 [44]). Unfortunately, they are not efficient enough.

The above considerations lead us to the following question:

Q3: *"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). Syntax of DMIPE is presented in Section 1.3.

1.2 Contributions

Below we list main results of our work.

- We introduce a novel primitives 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 construction of SE from DMIPE. As a result, we obtain a lattice-based SE, which is more efficient (in term 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 [28], FuPE [21], DFPE [22]. We point out that two versions of ADE (and hence PE, FuPE, DFPE) that can be easily built 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 alternatively from other secret keys by adding more vectors to the list of *predicate vectors*. More formally, DMIPE is defined by five main algorithms $\mathsf{DMIPE.Setup}$, $\mathsf{DMIPE.Derive}$, $\mathsf{DMIPE.Del}$, $\mathsf{DMIPE.Enc}$, $\mathsf{DMIPE.Dec}$. Given a vector space $\mathcal{D}$ that supports the inner product operation (e.g., $\mathcal{D} = \mathbb{Z}_q^d$ for q prime). For a security parameter λ and a setup parameter sp, the setup algorithm $\mathsf{DMIPE.Setup}(\lambda, \mathsf{sp})$ generates public parameters pp and a master secret key msk. The key generation algorithm $\mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$ takes public parameters pp, a master secret key msk and a list $\vec{V} \subset \mathcal{D}$ of vectors. It outputs a secret key $\mathsf{sk}_{\vec{V}}$ for $\vec{V}$. For public parameters pp, a secret key $\mathsf{sk}_{\vec{V}}$ for $\vec{V}$ and a predicate vector $\mathbf{v} \in \mathcal{D}$, the delegation algorithm $\mathsf{DMIPE.Del}(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathbf{v})$ returns a secret key for $\mathsf{sk}_{\vec{V}'}$ where $\vec{V}' = \vec{V} \cup \{\mathbf{v}\}$. The encryption algorithm $\mathsf{DMIPE.Enc}(\mathsf{pp}, \mu, \mathbf{x})$ outputs a ciphertext $\mathsf{ct}_{\mathbf{x}}$ for public parameters pp, a message/plaintext μ and an attribute vector $\mathbf{x} \in \mathcal{D}$. The decryption algorithm $\mathsf{DMIPE.Enc}(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathsf{ct}_{\mathbf{x}})$ either recovers the plaintext μ if $\langle \mathbf{x}, \mathbf{v} \rangle = 0$ for all $\mathbf{v} \in \vec{V}$ or returns $\perp$.

There is an important requirement for predicate vectors $\vec{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$. They have to be linearly independent, i.e. there is no vector that 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 $\vec{V}$.

Let us illustrate a possible application of DMIPE. Consider a company, where each officer/worker has a secret key allowing him/her to access internal documents. Assume that access is restricted depending on his/her role/department in the company. To this ends, each document is encrypted together with an attribute vector and each person in the company is issued with a predicate vector. A private key for each person corresponds to a list of predicate vectors. Furthermore, a manager can use her/his key to generate a key for subordinates using delegation.

In the work we show that DMIPE can be used to implement other primitives, e.g., SE, which in turn can be used to implement other primitives as previously mentioned. We argue that DMIPE is a generalization of IPE and it is more natural than HIPE. Compared to HIPE, 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).

Lattice-based DMIPE. At a high level description, our lattice-based DMIPE design exploits the lattice trapdoor mechanism and the lattice evaluation for inner product functions (see Lemma 1 and Lemma 2 for informal statements). First, we prepare by defining the inner product functions: For $\mathbf{v} \in \mathbb{Z}_q^d$, an inner product function indexed by a vector $\mathbf{v}$ is defined as $f_{\mathbf{v}}(\mathbf{x}) = \langle \mathbf{v}, \mathbf{x} \rangle \pmod q$, for all $\mathbf{x} \in \mathbb{Z}_q^d$.

Lemma 1 (Lattice Trapdoors, informal). *Given a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. There exists a procedure $\mathbf{A}_\sigma^{-1}$ that samples $\mathbf{A}_\sigma^{-1}(\mathbf{U}) \sim (\mathcal{D}_{\mathbb{Z}^m, \sigma})^{m'}$ in $\mathsf{poly}(n, m, \log q, m')$-*

Table 2: Comparison of IPE, HIPE and DMIPE

	IPE	HIPE	DMIPE
# Attribute vectors	1	d	1
# Predicate vectors	1	$\leq d$	≥ 1
Delegation?	No	Yes	Yes
Dimension of vectors	same	not necessarily same, depending on each vector's level in the hierarchical format	same

time for any $\mathbf{U} \in \mathbb{Z}_q^{n \times m'}$ *conditioned on* $\mathbf{A} \cdot \mathbf{A}_\sigma^{-1}(\mathbf{U}) = \mathbf{U}$. *One can always generate such a pair* $(\mathbf{A}, \mathbf{A}_\sigma^{-1})$ *by calling the* $\mathsf{TrapGen}$ *algorithm. Furthermore, using* $\mathbf{A}_\sigma^{-1}$, *one can compute* $[\mathbf{A}|\mathbf{B}]_\sigma^{-1}$, *or* $[\mathbf{B}|\mathbf{A}]_\sigma^{-1}$ *for any* $\mathbf{B}$. *We call* $\mathbf{A}_\sigma^{-1}$ *the* σ*-trapdoor for* $\mathbf{A}$.

Lemma 2 (Lattice Evaluation for Inner Product Functions, informal). *Let* $n, q, m = n\lceil \log q \rceil$ *be integers. There exists an efficient deterministic algorithm* $\mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}}, \mathbf{B})$ *that, on input a function* $f_{\mathbf{v}}$ *and a matrix* $\mathbf{B} \in \mathbb{Z}_q^{n \times md}$, *outputs a matrix* $\mathbf{H}_{\mathbf{v}} \in \{0,1\}^{md \times m}$ *such that* $\|\mathbf{H}_{\mathbf{v}}\|_{\max} \leq 1$ *and that for every* $\mathbf{x} \in \mathbb{Z}_q^d$, $[\mathbf{B} \pm \mathbf{x} \otimes \mathbf{G}]\mathbf{H}_{\mathbf{v}} = \mathbf{B}\mathbf{H}_{\mathbf{v}} \pm \langle \mathbf{v}, \mathbf{x} \rangle \cdot \mathbf{G} \pmod q$.

Now, we sketch the lattice-based DMIPE construction. For appropriately chosen parameters n, m, d, q, σ_0, a public key (included in the public parameters pp) consists of matrices $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times dm}$, $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$ and $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, which is generated together with a σ_0-trapdoor $\mathbf{A}_{\sigma_0}^{-1}$. The trapdoor acts as the master secret key.

To generate a key for a list $\vec{V}$ of k predicate vectors, we compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$ for each $\mathbf{v}_i \in \vec{V}$. Here, we use the public evaluation $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}_i}, \mathbf{B})$ from Lemma 2. The secret key $\mathsf{sk}_{\vec{V}}$ is the σ_0-trapdoor $\mathbf{A}_{\vec{V},\sigma_0}^{-1}$ for $\mathbf{A}_{\vec{V}}$ computed using the master secret key $\mathbf{A}_{\sigma_0}^{-1}$ via Lemma 7, where $\mathbf{A}_{\vec{V}} := [\mathbf{A}|\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$. Regarding delegation, given a secret key $\mathsf{sk}_{\vec{V}} := \mathbf{A}_{\vec{V},\sigma_0}^{-1}$, one can update it to get a secret key for $\vec{V}' := \vec{V} \cup \{\mathbf{v}_{k+1}\}$ for any $\mathbf{v}_{k+1}$. This can be done by generating a trapdoor $\mathbf{A}_{\vec{V}',\sigma_0}^{-1}$ for $\mathbf{A}_{\vec{V}'} = [\mathbf{A}_{\vec{V}}|\mathbf{B}_{\mathbf{v}_{k+1}}]$ from $\mathbf{A}_{\vec{V},\sigma_0}^{-1}$. Again, $\mathbf{B}_{\mathbf{v}_{k+1}}$ is computed by evaluating $(f_{\mathbf{v}_{k+1}}, \mathbf{B})$ using $\mathsf{EvalF}^{\mathsf{IP}}$.

Encryption on an attribute vector $\mathbf{x}$ employs the dual Regev's style to produce a ciphertext $\mathsf{ct}_{\mathbf{x}} := (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{mid}}, \mathbf{c}_{\mathsf{out}})$, in which $\mathbf{c}_{\mathsf{mid}} := \mathbf{s}^\top(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top \mathbf{R} \in \mathbb{Z}_q^{md}$. Here $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ is an LWE secret, $\mathbf{e}_{\mathsf{in}} \leftarrow \chi^m$ is an LWE error, and $\mathbf{R} \xleftarrow{\$} \{-1, 0, 1\}^{m \times md}$ is a high-entropy matrix. For decrypting a ciphertext $\mathsf{ct}_{\mathbf{x}}$ using a secret key $\mathsf{sk}_{\vec{V}}$, we again exploit Lemma 2 to compute $\mathbf{c}_{\mathbf{v}_i} := \mathbf{c}_{\mathsf{mid}}\mathbf{H}_{\mathbf{v}_i}$ for all $\mathbf{v}_i \in \vec{V}$. Putting $\mathbf{c}_{\mathbf{v}_i}$'s, $\mathbf{c}_{\mathsf{in}}$, and $\mathbf{c}_{\mathsf{out}}$ together allows us to recover the underlying plaintext if $\langle \mathbf{x}, \mathbf{v}_i \rangle = 0 \pmod q$ for all $\mathbf{v}_i \in \vec{V}$. Otherwise, decryption fails.

Equivalence of DMIPE and SE. We prove 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 that 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 [1,44], according to the generic framework of Chen et al. [18]. 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 predicate vectors)	**ct-size** (h attribute vectors, 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 [44] ($\Delta(d)$-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)	$(d+2)\cdot \mathbb{Z}_q^{n\times m}$	$1\cdot D_{\mathbb{Z}}^{m\times m}$	$1\cdot D_{\mathbb{Z}}^{km\times m}$	$(d+2)\cdot \mathbb{Z}_q^{m}$

Security Notions of DMIPE and SE. There are several security notions for DMIPE and SE, which are the same as for PrE. First, we recap security notions for PrE as introduced in the work [31]. Then, we discuss notions that are relevant for DMIPE and SE only.

S1. **Payload-hiding.** It is the most basic security notion for PrE. It ensures that a ciphertext leaks no information about the underlying plaintext (but attributes can be revealed). In the challenge phase, an adversary must submit an attribute, say $\mathbf{x}^*$ (together with two plaintexts μ_0^*, μ_1^* of the same length). Any key queries related to predicate f subject to the restriction $f(\mathbf{x}^*) = 0$.

S2. **Attribute-hiding.** This security notion is stronger than payload-hiding. It requires from ciphertext to leak no information about either plaintext or attached attributes. When challenged, an adversary must submit two attributes $\mathbf{x}_0^*, \mathbf{x}_1^*$ (together with two plaintexts μ_0^*, μ_1^* of the same length). Any key queries related to predicate f are subject to the restriction that $f(\mathbf{x}_0^*) = f(\mathbf{x}_1^*)$. If there is a query related to some predicate f such that $f(\mathbf{x}_0^*) = f(\mathbf{x}_1^*) = 1$, then it is required that $\mu_0^* = \mu_1^*$. There is an intermediate notion called *weak attribute-hiding* [4]. This notion requires that $f(\mathbf{x}_0^*) = 0 \wedge f(\mathbf{x}_1^*) = 0$ for any key query on predicates f's.

S3. **Selective security.** For this notion, an adversary must commit to its target attribute(s) ahead of time before seeing any keys.

S4. **Adaptive (full) security.** For this notion, an adversary does not have to commit to its target attribute(s) until the challenge phase starts.

We can combine S1/S2 with S3/S4 notions and get the following four security notions for PrE (and hence DMIPE, SE and even ADE): 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. Note that our lattice-based DMIPE actually enjoys the selective payload-hiding se-

curity. This is the result of our assumption that the decisional LWE problem is intractable.

ADE and the Construction from SE. ADE is parametrized by a security parameter λ, a maximum number of negative tags per ciphertext $d = d(\lambda)$ and a maximum number of positive tags $a = a(\lambda)$. Now, consider an allow list AL and a deny list DL. AL consists of positive tags, while DL holds negative tags.

ADE is defined by its five algorithms, namely: *key generation,* $(\mathsf{pp}, \mathsf{sk}^{\emptyset}_{\emptyset}) \leftarrow \mathsf{ADE.Gen}(1^{\lambda}, 1^{a}, 1^{d})$; *positive puncturing,* $\mathsf{sk}^{AL'_1 \cup AL'_2}_{DL'} \leftarrow \mathsf{ADE.Ppun}(\mathsf{pp}, \mathsf{sk}^{AL'_1}_{DL'}, AL'_2, k)$, here k is only used in the k-tADE variant; *negative puncturing,* $\mathsf{sk}^{AL'}_{DL'_1 \cup DL'_2} \leftarrow \mathsf{ADE.Npun}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'_1}, DL'_2)$; *encryption,* $\mathsf{ct}^{AL}_{DL} \leftarrow \mathsf{ADE.Enc}(\mathsf{pp}, \mu, AL, DL)$; and *decryption,* $\mu/\perp \leftarrow \mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}^{AL'}_{DL'}, \mathsf{ct}^{AL}_{DL})$. Note that, positive puncturing and negative puncturing can be done in any order.

We consider three versions of ADE: (i) standard ADE (sADE); (ii) inclusive ADE (iADE) and (iii) k-threshold ADE (k-tADE). For all three ADE versions, it is required that the initial key $sk^{\emptyset}_{\emptyset}$ can successfully decrypt any ciphertexts. However, when punctured, in order for $\mathsf{sk}^{AL'}_{DL'}$ to be able to successfully decrypt ct^{AL}_{DL}): for sADE, it requires $((AL = AL') \wedge (DL \cap DL' = \emptyset))$; for iADE, it requires $((AL' \subseteq AL) \wedge (DL \cap DL' = \emptyset))$; for k-tADE, it requires $((|AL \cap AL'| \geq k) \wedge (DL \cap DL' = \emptyset))$.

We can construct sADE and iADE from SE by applying the following encodings. For key generation, an allow list $AL' = \{p_1, \cdots, p_k\}$ is associated with space $W_{AL'} := \{(p_1, \cdots, p_k, x_{k+1}, \cdots, x_a)^{\top} : x_i \in \mathbb{Z}_q\} \subseteq \mathbb{Z}_q^a$; while a deny list DL', is connected to space $W_{DL'} := \mathsf{span}\{\mathbf{v}_x : x \in DL''\}$. Here, $DL'' := \mathcal{T}^{(-)} \setminus DL'$ and $\mathbf{v}_x := (1, x, x^2, \cdots, x^{2d-1})$ is a Vandermonde vector. The predicate space related to $\mathsf{sk}^{AL'}_{DL'}$ is $W_{\mathsf{key}} := W_{AL'} \times W_{DL'}$. For encryption, allow list $AL = \{p_1, \cdots, p_k\}$ is sticky to vector $\mathbf{x}_{AL} := (p_1, \cdots, p_k, 0, \cdots, 0) \in \mathbb{Z}_q^a$; while deny list DL is translated into vector $\mathbf{x}_{DL} := \sum_{x \in DL} \mathbf{v}_x \in \mathbb{Z}_q^{2d}$. The attribute vector corresponding to ct^{AL}_{DL} is $\mathbf{x}_{\mathsf{ct}} := (\mathbf{x}_{AL}, \mathbf{x}_{DL}) \in \mathbb{Z}_q^{a+2d}$. Note that, spaces mentioned here are possibly affine ones. More details are given in Section 7. However, how to translate k-tADE to SE is an open problem.

1.4 Related Works

Katz, Sahai, and Waters [31] have proposed IPE together with the generalized concept of PrE. They have designed a PrE scheme for inner products over $\mathbb{Z}_N$ for some large integer N. They have introduced the notion of payload-hiding and attribute-hiding for generic predicates as well as for the inner product predicate.

Okamoto and Takashima [35] have introduced the concept of *hierarchical predicate encryption for inner products* (HPE), which is called HIPE in this work. They have deployed the notion of dual pairing vector spaces (DPVS) to design their HIPE, whose security is proven assuming intractability of the decisional subspace problem (DSP) [34]. As mentioned above, the first lattice-based HIPE has been constructed by Abdalla et al. [1]. The construction has been improved by Xagawa [44].

Agrawal et al. [4] have proposed the first lattice-based IPE. The scheme has been used by Abdalla et al. [1] to construct their lattice-based HIPE. Security of both IPE and HIPE is proven assuming intractability of the decisional LWE

problem. In particular, IPE security is weakly attribute hiding [4]. Unfortunately, both IPE and HIPE are impractical (see [1, 4]).

Lattice evaluation algorithms have been proposed and developed in a long sequence of works [3], [33] [24], [7] [9], with many modifications and abstractions (for example, see [26], [15]). The technique is being used to realize many important primitives for lattice manipulation. The reader is referred to works [11], [14], [37], [41] for details. Recently, Tsabary [42] has exploited lattice evaluation algorithms together with constrained pseudorandom functions (CPRF) to construct a fully secure ABE for t-CNF from LWE. Katsumata et al. [30] have used the techniques from [42] to relax the "conforming" condition of CPRF. Katsumata et al. have taken advantage of a specific linearity property in lattice evaluation algorithms. As the result, they have been able to construct the first LWE-based payload-hiding adaptively secure IPE over $\mathbb{Z}$ (but can be converted to one over $\mathbb{Z}_q$ for a polynomial-size q). We believe that it is possible to employ the work [30] to construct a *adaptively secure* DMIPE in the lattice setting. It is clear, however, that design of a delegation mechanism might be the main challenge in this case. We leave this problem for a future research.

1.5 Organisation

The rest of the paper is organized as follows. Section 2 presents preliminaries, in particular, it gives the syntax and security of SE together with a mathematical background necessary for the reader to follow our lattice-based DMIPE design. Section 3 introduces a novel primitive DMIPE. Section 4 proposes a generic SE construction from DMIPE. We then present a concrete lattice-based DMIPE design in Section 5, whose security depends on intractability of the decision variant of the learning with errors problem. The section also details our security proofs and gives a guide for parameter selection. Section 6 gives the generic reduction from SE to DMIPE. By combining results of both Sections 4 and 6, it is possible to establish the equivalence between SE and DMIPE. In Section 7, we formally state the definition and security notions for ADE and then show how to translate some of ADE variants to SE. Finally, Section 8 concludes the work and discusses possible future research directions.

2 Preliminaries

Notation. Given a discrete set S, we denote its cardinality by $|S|$. 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, a vector is represented by a small bold-face letter, e.g., $\mathbf{x}$ and in the column form unless stated otherwise. A matrix is written as a capital bold-face letter, 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. We sometimes write $\mathbf{A} \sim \chi$ to say $\mathbf{A}$ follows the distribution χ, while $\mathbf{A} \leftarrow \chi$ is used to say that $\mathbf{A}$ is sampled from the distribution χ. The notation $\mathbf{A} \overset{\mathsf{negl}}{\sim} \chi$ means that the distribution of $\mathbf{A}$ is negligibly close to χ. $\mathbf{A} \xleftarrow{\$} X$ says that $\mathbf{A}$

is sampled uniformly at random from the set X. A function negligible in n is denoted as $\mathsf{negl}(n)$, while a function which is a polynomial in n is denoted as $\mathsf{poly}(n)$. Finally, all logarithms are for base 2.

2.1 Framework of Spatial Encryption

Boneh and Hamburg [10, 29] have introduced the Spatial Encryption (SE). SE belongs to PrE ans also to a bigger class called *generalized identity-based encryption* (GIBE). Cryptographic operations of GIBE are controlled by *role* and *policy* components. Encryption takes a plaintext together with a role and produces a ciphertext. Decryption key is derived using a policy. If the role and policy fulfil a certain condition, then a plaintext hidden in a ciphertext can be recovered. Otherwise, decryption fails. Consider SE, then a role is an affine (or vector) subspace V of the *top* space T. A policy is an affine point (or a vector) $\mathbf{v}$ that belongs to the top space T. The condition $\mathbf{v} \in V$ has to hold for successful decryption. Additionally, SE allows to delegate decryption. This means that a decryption key for a subspace $V_1 \sqsubseteq \mathsf{T}$ should allow to get a decryption key for $V_2 \sqsubseteq V_1$. If roles and policies are affine points/subspaces, we call such SE *affine*. It they are vectors/subspaces, we call it *linear*. The dimension of SE is the corresponding dimension of the top space.

Syntax. Let us recall the syntax of SE. Formally, SE 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 λ and setup parameters sp. It returns public parameters pp which implicitly defined a top space T and a master secret key msk. The master key msk can be seen as the secret key sk_T (i.e., $\mathsf{msk} = \mathsf{sk}_\mathsf{T}$) for the top space T.

$\mathsf{sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$**:** The algorithm takes as input the master secret key msk and a subspace V. It outputs the secret key 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 sk_{V_1} for the space V_1. It outputs the secret key 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 μ under a point/vector $\mathbf{x}$. It outputs a ciphertext $\mathsf{ct}_\mathbf{x}$.

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

Correctness. It requires that for all λ, 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 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 delegation from another subspace).

Security Notions for SE. As discussed before, SE security notions include selective/adaptive payload/attribute-hiding. However, we only formally define

$\mathsf{SE}^{\mathsf{sel},\mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}$ **Game** ($\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$) :

- $\mathbf{x}^* \leftarrow \mathcal{A}(1^\lambda, \mathsf{sp})$;
- $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{SE.Setup}(1^\lambda, \mathsf{sp})$, $\mathsf{VKList} \leftarrow \emptyset$; // VKList is to store (V, sk_V)
- $(\mu_0^*, \mu_1^*) \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\mathsf{KD}(\cdot,\cdot),\mathsf{DQ}(\cdot,\cdot)}(\mathsf{pp})$;
- $b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}^*_{\mathbf{x}^*} \leftarrow \mathsf{SE.Enc}(\mathsf{pp}, \mathbf{x}^*, \mu_b^*)$;
- $b' \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\mathsf{KD}(\cdot,\cdot),\mathsf{DQ}(\cdot,\cdot)}(\mathsf{pp}, \mathsf{ct}^*_{\mathbf{x}^*})$. // **Restrictions:** Not allowed $\mathsf{DQ}(V, \mathsf{ct}^*_{\mathbf{x}^*})$ with $\mathbf{x}^* \in V$.

Queried Oracles:

- $\mathsf{KQ}(\mathrm{V})$ (allowed only if $\mathbf{x}^* \notin V$): Take sk_V from $(V, \mathsf{sk}_V) \in \mathsf{VKList}$ (if exists there). Otherwise, run $\mathsf{sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$. Update $\mathsf{VKList} \leftarrow \mathsf{VKList} \cup \{(V, \mathsf{sk}_V)\}$.
- $\mathsf{KD}(V_1, V_2)$ (allowed only if $V_2 \sqsubseteq V_1 \wedge \mathbf{x}^* \notin V_1$): Take sk_{V_1} from $(V_1, \mathsf{sk}_{V_1}) \in \mathsf{VKList}$ (if exists there). Otherwise, run $\mathsf{sk}_{V_1} \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V_1)$ and return $\mathsf{sk}_{V_2} \leftarrow \mathsf{SE.Del}(\mathsf{pp}, \mathsf{sk}_{V_1}, V_2)$. Update $\mathsf{VKList} \leftarrow \mathsf{VKList} \cup \{(V_i, \mathsf{sk}_{V_i})\}$ for $i = 1, 2$.
- $\mathsf{DQ}(V, \mathsf{ct}_{\mathbf{x}})$ (allowed only if $\mathsf{ATK} = \mathsf{CCA}$): Take sk_V from $(V, \mathsf{sk}_V) \in \mathsf{VKList}$ (if exists there). Otherwise, run $\mathsf{sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$, then return the output of $\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$. Update $\mathsf{VKList} \leftarrow \mathsf{VKList} \cup \{(V, \mathsf{sk}_V)\}$.

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

the selective payload-hiding security for SE – see Definition 1 and the game $\mathsf{SE}^{\mathsf{sel},\ \mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}$ in Figure 2. Note that $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$, where CPA and 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* $\mathsf{SE}^{\mathsf{sel},\mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}$ *game is negligible, i.e.,* $\mathsf{Adv}^{\mathsf{payload},\mathsf{ATK}}_{\mathsf{SE},\mathcal{A},\mathsf{sel}} := |\Pr[b' = b] - 1/2| = \mathsf{negl}(\lambda)$.

2.2 Lattices, Gaussians, Trapdoors

Norms. In this paper, 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 $\mathbf{A} = (a_{i,j})^{j\in[m]}_{i\in[n]}$, $\|\mathbf{a}\|_{\max} := \max_{i\in[n]} |a_i|$ 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 3. *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,*

1. $\|\mathbf{e}_1^\top \mathbf{A}_1\|_{\max} \le m_1 \|\mathbf{e}_1^\top\|_{\max} \cdot \|\mathbf{A}_1\|_{\max}$.
2. $\|(\mathbf{e}_2^\top | \mathbf{e}_3^\top)\mathbf{A}_2\|_{\max} \le (m_2\|\mathbf{e}_2^\top\|_{\max} + m_3\|\mathbf{e}_3^\top\|_{\max}) \cdot \|\mathbf{A}_2\|_{\max}$.

Lattices. An integer lattice can always be represented as a set $\mathcal{L} = \mathcal{L}(\mathbf{B}) := \{\mathbf{Bx} : \mathbf{x} \in \mathbb{Z}^m\} \subseteq \mathbb{Z}^n$, where $\mathbf{B} \in \mathbb{Z}^{n\times m}$ is a basis for the lattice $\mathcal{L}$. For $\mathbf{A} \xleftarrow{\$} \mathbb{Z}^{n\times m}$, $\mathbf{u} \in \mathbb{Z}_q^n$ and $\mathbf{U} \in \mathbb{Z}_q^{n\times k}$, we focus on the following lattices: $\Lambda_q^\perp(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m \mid \mathbf{Ae} = 0 \pmod q\}$, $\Lambda_q^{\mathbf{u}}(\mathbf{A}) := \{\mathbf{e} \in \mathbb{Z}^m | \mathbf{Ae} = \mathbf{u} \pmod q\}$ and $\Lambda_q^{\mathbf{U}}(\mathbf{A}) := \{\mathbf{R} \in \mathbb{Z}^{m\times k} | \mathbf{AR} = \mathbf{U} \pmod q\}$. Note that, $\mathbb{Z}^m$ for any $m \in \mathbb{N}$ is also a lattice.

[5] Some papers (e.g., [15], [42], [30]) denote this max-absolute-value norm by $\|\cdot\|_\infty$.

Distributions. We sample lattice vectors using a discrete Gaussian distribution to keep their size sufficiently short. Given $n \geq 1$, a lattice $\mathcal{L} \subseteq \mathbb{Z}^n$, a vector $\mathbf{v} \in \mathbb{R}^n$, a real number $\sigma > 0$. Then, a discrete Gaussian distribution over $\mathcal{L}$ centred at $\mathbf{v}$ with (Gaussian) parameter σ 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 $D_{\mathcal{L},\sigma}$. The following lemma says how short a vector sampled via a discrete Gaussian distribution (over $\mathbb{Z}$) is.

Lemma 4 ([32, Lemma 4.4]). $\Pr[|x| > k\sigma : \ x \leftarrow D_{\mathbb{Z},\sigma}] \leq 2\exp(-\frac{k^2}{2})$.

Note that, in Lemma 4, if we set $k = 12$, then $\Pr[|x| \leq 12\sigma : \ x \leftarrow D_{\mathbb{Z},\sigma}] \geq 1 - 2\exp(-72) \approx 1 - 2^{-100}$. We also consider the (B,ϵ)-bounded distributions χ supported over $\mathbb{Z}$ which is required that $\Pr[|x| > B : x \leftarrow \chi] \leq \epsilon$. In this sense, $\chi = D_{\mathbb{Z},\sigma}$ is a $(12\sigma, 2^{-100})$-bounded distribution.

Randomness Extraction. 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 5 (Leftover Hash Lemma, [2, Lemma 13]). *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 $\mathbf{e} \in \mathbb{Z}_q^m$.*

Decisional Variant of Learning with Errors (DLWE). DLWE is defined below and is used as our intractability assumption to prove security of our DMIPE design.

Definition 2 (DLWE, [38]). *An instance of the decisional learning with errors problem (n, m, q, χ)-DLWE is parametrized by positive integers n, m, a prime q and a distribution χ over $\mathbb{Z}_q$. The advantage of a probabilistic polynomial time (PPT) distinguisher $\mathcal{S}$ in solving the (n, m, q, χ)-DLWE problem is defined by the difference of probabilities in distinguishing the two joint distributions $(\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top)$ and $(\mathbf{A}, \mathbf{c}^\top)$, where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n\times m}, \mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n, \mathbf{e} \leftarrow \chi^m, \mathbf{c} \xleftarrow{\$} \mathbb{Z}_q^m$. Formally, $\mathbf{Adv}_{\mathcal{S}}^{(n,m,q,\chi)-\mathsf{DLWE}} := |\Pr[\mathcal{S}(\mathbf{A}, \mathbf{s}^\top\mathbf{A} + \mathbf{e}^\top) = 1] - \Pr[\mathcal{S}(\mathbf{A}, \mathbf{c}^\top) = 1]|$. The (n, m, q, χ)-DLWE assumption holds if $\mathbf{Adv}_{\mathcal{S}}^{(n,m,q,\chi)-\mathsf{DLWE}} \leq \mathsf{negl}(n)$ for all $\mathcal{S}$.*

The hardness of the DLWE problem is guaranteed by a quantum reduction [38] or classical reduction [36], [13] to the worst-case GapSVP problem to within a factor of $n^{O(d)}$ if we take χ is (B,ϵ)-bounded, with $B = q/n^d$; see [9] for discussion. We restate Lemma 6 followed from [15] for choosing parameters in Section 5.

Lemma 6 (DLWE Hardness, [15, Corollary 3.2]). *Given the (n, m, q, χ)-DLWE problem, then it is at least as hard as the classical $GapSVP_\gamma$ and the quantum $SIVP_\gamma$, where $q = q(n) \leq 2^n$, $m = \Theta(n\log q) = \mathsf{poly}(n)$, $\chi = \chi(n)$ such that χ is a (B,ϵ)-bounded for some $B = B(n)$, $q/B \geq 2^{n^\epsilon}$ and $\gamma = 2^{\Omega(n^\epsilon)}$.*

Gadget Matrix. Let n, m, q be positive integers, and $m \geq n\lceil \log q \rceil$. The gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ is defined by taking the tensor product $\mathbf{I}_n \otimes (1, 2, \cdots, 2^{\lceil \log q \rceil - 1})$ then padding $(m - n\lceil \log q \rceil)$ zero columns. In addition, for any $k \in \mathbb{N}$, there exists an efficient deterministic algorithm $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times k} \to \{0,1\}^{m \times k}$ such that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times k}$, $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

Lattice Trapdoors. We follow the works [15], [42], [30]. In particular, let n, m, q be positive integers and consider a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$. For any positive integer m' and any matrix $\mathbf{U}$ (including zero matrices) in $\mathbb{Z}_q^{n \times m'}$, we denote by $\mathbf{A}_\sigma^{-1}(\mathbf{U})$ a random variable that is drawn from a discrete Gaussian distribution $(D_{\mathbb{Z}^m,\sigma})^{m'}$ provided $\mathbf{A} \cdot \mathbf{A}_\sigma^{-1}(\mathbf{U}) = \mathbf{U}$. Now, a σ-trapdoor is defined as a procedure that enables us to sample $\mathbf{A}_\sigma^{-1}(\mathbf{U})$ in polynomial time in $n, m, \log q, m'$ for any $\mathbf{U}$. With a slight notational abuse, we also take $\mathbf{A}_\sigma^{-1}$ to denote the σ-trapdoor for $\mathbf{A}$. In particular, it is shown in [33] that the gadget matrix $\mathbf{G}$ has a publicly known constant trapdoor. In other works, it is often denoted as $\mathbf{T}_\mathbf{G} \in \mathbb{Z}^{m \times m}$. In our work, we denote it as $\mathbf{G}_{O(1)}^{-1}$. We briefly summarise some standard results and algorithms for handling lattice trapdoors that we use in our design. The writing style we use follows [15], [42], [30].

Lemma 7 ([5], [23], [8], [2], [16], [33]). *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 $\mathbf{A}_{\sigma_0}^{-1}$ 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 $\mathbf{A}_{\sigma_1}^{-1}$, one can compute $\mathbf{A}_{\sigma_2}^{-1}$ for any $\sigma_2 \geq \sigma_1$.*
3. *Given a trapdoor $\mathbf{A}_\sigma^{-1}$, one can compute $[\mathbf{A}|\mathbf{B}]_\sigma^{-1}$, $[\mathbf{B}|\mathbf{A}]_\sigma^{-1}$ for any matrix $\mathbf{B}$ having the same number of rows as $\mathbf{A}$.*
4. *Given the gadget matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ defined above, using its trapdoor $\mathbf{G}_{O(1)}^{-1}$ one can compute the trapdoor $[\mathbf{A}|\mathbf{AR} + \mathbf{G}]_\sigma^{-1}$ for all $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{R} \in \mathbb{Z}^{n \times m'}$ with $m' > n\lceil \log q \rceil$ for $\sigma = m \cdot \|\mathbf{R}\|_{\max} \cdot \omega(\sqrt{\log m})$.*
5. *For a trapdoor $\mathbf{A}_{\sigma_1}^{-1}$ and for any $\mathbf{U} \in \mathbb{Z}_q^{n \times m'}$, by Lemma 4, $\Pr[\|\mathbf{A}_{\sigma_1}^{-1}(\mathbf{U})\|_{\max} \leq 12\sigma : \mathbf{x} \leftarrow D_{\mathbb{Z},\sigma}] \geq 1 - 2^{-100}$.*

2.3 Lattice Evaluation for Inner Product Functions

Boneh et al. [9] have introduced algorithms that allow to evaluate unbounded fan-in arithmetic circuits of a polynomial depth. This technique has arisen from the sequence of works by Miciancio and Peikert [33], Gentry et al. [24] and Boneh et al. [9]. A further progress is due to works by Brakerski and Vaikuntanathan [15], Brakerski et al. [14] and Tsabary [42]. Note that the works [15], [14], [42] consider binary functions, i.e., $f : \{0,1\}^* \to \{0,1\}^*$ rather than functions over $\mathbb{Z}_q$.

In this work, we only focus on a family of inner product functions defined over $\mathbb{Z}_q$. Specifically, for every $\mathbf{x} \in \mathbb{Z}_q^d$, an inner product function $f_\mathbf{v} : \mathbb{Z}_q^n \to \mathbb{Z}_q$ 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, [42, Theorem 2], for example, is not suitable for our work. Besides, we cannot use the

norm bound of $\widehat{\mathbf{H}}$ and $\mathbf{H}$ as in [42, Theorem 2] since they are quite large. Recall that inner product functions defined over $\mathbb{Z}_q$ can be represented as an addition gate [9, Section 4.2] whose depth is $\ell = 1$. The following lemma is sufficient for our purpose.

Lemma 8 (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. 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, $\|\mathbf{H}\|_{\max} \leq 1$ as $\mathbf{H} \in \{0,1\}^{md \times m}$. □

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 $\mathbb{Z}_q$. As we mentioned in Section 1.3, the set of predicate vectors $\vec{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$ embedded in a decryption key is always linearly independent. This requirement is natural because otherwise some vectors are redundant. Moreover, this guarantees that the delegation algorithm is well-defined in the following sense. A delegated decryption key for $\vec{V} \cup \mathbf{v}$ can only be issued by someone who has a secret key for $\vec{V}$ and $\mathbf{v}$ is linearly independent of the existing predicate vectors from $\vec{V}$.

Syntax of DMIPE. A DMIPE consists of the five algorithms DMIPE.Setup, DMIPE.Derive, DMIPE.Del, DMIPE.Enc and DMIPE.Dec. They are formally defined below.

$(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathbf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$**:** The algorithm takes as input a security parameter λ and a setup parameters sp. It returns public parameters pp and a master secret key msk.

$\mathsf{sk}_{\vec{V}} \leftarrow \mathbf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$**:** The algorithm takes a master secret key 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 $\vec{V}$.

$\perp$ /$\mathsf{sk}_{\vec{V}_2} \leftarrow$ **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 $\perp$.

$\mathsf{ct}_{\mathbf{x}} \leftarrow$ **DMIPE.Enc**$(pp, \mu, \mathbf{x})$**:** The algorithm encrypts a message μ under a vector $\mathbf{x}$ and produces a ciphertext $ct_{\mathbf{x}}$.

$\perp/\mu \leftarrow$ **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. It requires that:
For all λ, 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 3 and Figure 3 together describe our security notion.

Definition 3. *DMIPE is selective payload-hiding secure if the advantage of the adversary playing in the* $\mathsf{DMIPE}^{\mathsf{sel,ATK}}_{\mathsf{payload},\mathcal{A}}$ *game (in Figure 3) is negligible, i.e.,*

$$\mathsf{Adv}^{\mathsf{payload,ATK}}_{\mathsf{DMIPE},\mathcal{A},\mathsf{sel}} := |\Pr[b' = b] - 1/2| = \mathsf{negl}(\lambda).$$

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 the affine/linear algebra.

4.1 Selected Facts from 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 $d+1$. Recall that for linear SE, we encrypt a plaintext together with a vector $\mathbf{x}$ and a decryption key is produced using

$\mathsf{DMIPE}^{\mathsf{sel,ATK}}_{\mathsf{payload},\mathcal{A}}$ **Game** ($\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$) :

- $\mathbf{x}^* \leftarrow \mathcal{A}(1^\lambda, \mathsf{sp})$;
- $(\mathsf{pp}, \mathsf{msk}) \leftarrow \mathsf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$, $\mathsf{VKList} \leftarrow \emptyset$; // VKList is to store $(\vec{V}, \mathsf{sk}_{\vec{V}})$
- $(\mu_0^*, \mu_1^*) \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\mathsf{KD}(\cdot,\cdot),\mathsf{DQ}(\cdot,\cdot)}(\mathsf{pp})$;
- $b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}^*_{\mathbf{x}^*} \leftarrow \mathsf{DMIPE.Enc}(\mathsf{pp}, \mathbf{x}^*, \mu_b^*)$;
- $b' \leftarrow \mathcal{A}^{\mathsf{KQ}(\cdot),\mathsf{KD}(\cdot,\cdot),\mathsf{DQ}(\cdot,\cdot)}(\mathsf{pp}, \mathsf{ct}^*_{\mathbf{x}^*})$. // **Restrictions:** Not allowed $\mathsf{DQ}(\vec{V}, \mathsf{ct}^*_{\mathbf{x}^*})$ with $\mathbf{x}^* \perp \vec{V}$.

Queried Oracles:

- $\mathsf{KQ}(\vec{V})$ (allowed only if $\mathbf{x}^* \not\perp \vec{V}$): Take $\mathsf{sk}_{\vec{V}}$ from $(\vec{V}, \mathsf{sk}_{\vec{V}}) \in \mathsf{VKList}$ (if exists there). Otherwise, run $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$. Update $\mathsf{VKList} \leftarrow \mathsf{VKList} \cup \{(\vec{V}, \mathsf{sk}_{\vec{V}})\}$.
- $\mathsf{KD}(\vec{V}_1, \mathbf{v})$ (allowed only if $\mathbf{v} \notin \vec{V}_1 \wedge \mathbf{x}^* \not\perp \vec{V}_1$): Take $\mathsf{sk}_{\vec{V}_1}$ from $(\vec{V}_1, \mathsf{sk}_{\vec{V}_1}) \in \mathsf{VKList}$ (if exists there). Otherwise, run $\mathsf{sk}_{\vec{V}_1} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V}_1)$ and return $\mathsf{sk}_{\vec{V}_2} \leftarrow \mathsf{DMIPE.Del}(\mathsf{pp}, \mathsf{sk}_{\vec{V}_1}, \mathbf{v})$. Here $\vec{V}_2 = \vec{V}_1 \cup \{\mathbf{v}\}$. Update $\mathsf{VKList} \leftarrow \mathsf{VKList} \cup \{(\vec{V}_i, \mathsf{sk}_{\vec{V}_i})\}_{i=1,2}$.
- $\mathsf{DQ}(V, \mathsf{ct}_{\mathbf{x}})$ (allowed only if ATK=CCA): Take $\mathsf{sk}_{\vec{V}}$ from $(V, \mathsf{sk}_{\vec{V}}) \in \mathsf{VKList}$ (if exists there). Otherwise, run $\mathsf{sk}_{\vec{V}} \leftarrow \mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$, then return the output of $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_{\vec{V}})$. Update $\mathsf{VKList} \leftarrow \mathsf{VKList} \cup \{(\vec{V}, \mathsf{sk}_{\vec{V}})\}$.

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 $\mathbf{x} \not\perp \vec{V}$.

a vector space V. Successful decryption using the decryption key requires that $\mathbf{x} \in V$. We need a tool that helps us to 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 9 ([20, Algorithm 2.3.7] and [18]). *There exists an efficient algorithm, named OCB, such that on input a vector space V outputs a basis, named $\mathcal{B}^\perp(V)$, for the orthogonal complement $V^\perp$ of V. Furthermore, the algorithm guarantees that if $V_2 \sqsubseteq V_1$ then $\mathcal{B}^\perp(V_1) \subseteq \mathcal{B}^\perp(V_2)$.*

4.2 Construction

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 a 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 λ, a system parameters 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 pp, the master secret key msk and a subspace V, perform:

1. Run $\mathcal{B}^\perp(V) \leftarrow \mathsf{OCB}(V)$, and set $\vec{V} := \{\mathbf{v} : \mathbf{v} \in \mathcal{B}^\perp(V)\}$.
2. 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 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:

1. 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 $V_2 \sqsubseteq V_1$, $\vec{V}_1 \subseteq \vec{V}_2$.
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 $\vec{V} \leftarrow \vec{V} \cup \{\mathbf{v}_i\}$.
3. At this point, we reach $\vec{V} = \vec{V}_2$. Finally, output $\mathsf{sk}_{V_2} := \mathsf{dmipe.sk}_{\vec{V}_2}$.

By doing this, it is clear that the distribution of the private keys are independent of the path taken. Namely, the distribution for the key sk_{V_3} computed from sk_{V_2} is the same as that of sk_{V_3} computed from sk_{V_1} with $V_3 \sqsubseteq V_2 \sqsubseteq V_1$.

SE.Enc$(\mathsf{pp}, \mathbf{x}, \mu)$**:** For input the public parameters pp, an attribute vector $\mathbf{x}$ and a plaintext μ, run $\mathsf{dmipe.ct}_{\mathbf{x}} \leftarrow \mathsf{DMIPE.Enc}(\mathsf{pp}, \mathbf{x}, \mu)$ and output a ciphertext $\mathsf{ct}_{\mathbf{x}} := \mathsf{dmipe.ct}_{\mathbf{x}}$.

SE.Dec$(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$**:** For input the public parameters pp, a ciphertext $\mathsf{ct}_{\mathbf{x}}$ and a secret key sk_V for a space V, return the output of $\mathsf{DMIPE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$.

The correctness of SE is established by Theorem 1.

Theorem 1. *The SE Π_{SE} is correct assuming correctness of the underlying DMIPE Π_{DMIPE}.*

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

4.3 Security Proof

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

Proof. The adversary $\mathcal{A}$ will take the role of the SE challenger playing with $\mathcal{S}$. And, the winning strategy of $\mathcal{A}$ is to simulate the environment of the same security game for Π_{SE} for the SE adversary $\mathcal{S}$ to join. The reduction is described below.

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

Query 1. For any query receiving from $\mathcal{S}$, $\mathcal{A}$ first uses the algorithm OCB to converts the queries into the forms compatible with DMIPE, 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 the 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 $\mathcal{S}$. □

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 this function can be represented as an addition gate; see [9, Section 4].

Our lattice-based DMIPE construction exploits the lattice trapdoor mechanism [23] [2] [33] and the lattice evaluation algorithms developed in a long series of works [33], [24], [9], [42].

We set up the public key (included in public parameters pp) by sampling $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times dm}$ and generating $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$ together with a σ_0-trapdoor $\mathbf{A}_{\sigma_0}^{-1}$ using $\mathsf{TrapGen}$. The trapdoor $\mathbf{A}_{\sigma_0}^{-1}$ is now the master secret key.

To generate a key on a list $\vec{V}$ of predicate vectors, for each $\mathbf{v}_i \in \vec{V}$, we evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}_i}, \mathbf{B})$ (Lemma 8) from which will get the corresponding matrix $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$. Now, the secret key $\mathsf{sk}_{\vec{V}}$ for $\vec{V}$ is a trapdoor $\mathbf{A}_{\vec{V},\sigma_0}^{-1}$ for $\mathbf{A}_{\vec{V}} := [\mathbf{A}|\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$ where $k = |\vec{V}|$, which can be easily produced using the master secret key $\mathbf{A}_{\sigma_0}^{-1}$ (thanks to Lemma 7). Performing in the same way, from $\mathbf{A}_{\vec{V},\sigma_0}^{-1}$ we can also delegate a key for any $\vec{V}'$ satisfying $\vec{V} \subseteq \vec{V}'$. Encryption on a message produces a ciphertext of dual Regev's style consisting of $\mathsf{ct}_{\mathbf{x}} := (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{mid}}, \mathbf{c}_{\mathsf{out}})$, with $\mathbf{c}_{\mathsf{mid}} := \mathbf{s}^\top(\mathbf{B} - \mathbf{x} \otimes \mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top \mathbf{R} \in \mathbb{Z}_q^{md}$. Here $\mathbf{x}$ is the attribute vector, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$ is an LWE secret, $\mathbf{e}_{\mathsf{in}} \leftarrow \chi^m$ is an LWE error, and $\mathbf{R} \xleftarrow{\$} \{-1, 0, 1\}^{m \times md}$ is a high-entropy matrix. Decrypting a ciphertext $\mathsf{ct}_{\mathbf{x}}$ using a secret key $\mathsf{sk}_{\vec{V}}$, needs to compute $\mathbf{c}_{\mathbf{v}_i} := \mathbf{c}_{\mathsf{mid}}\mathbf{H}_{\mathbf{v}_i}$ for all $\mathbf{v}_i \in \vec{V}$. Combining $\mathbf{c}_{\mathbf{v}_i}$'s, $\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{out}}$ together allows us to recover the underlying plaintext if and only if $\langle \mathbf{x}, \mathbf{v}_i \rangle = 0 \pmod q$ for all $\mathbf{v}_i \in \vec{V}$.

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

$\mathsf{DMIPE.Setup}(1^\lambda, 1^d)$: On input a security parameter λ, a dimension d, do the following:

1. Choose n, m, q according to λ, d. Also, choose a (B, ϵ)-bounded distribution χ 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.
2. Choose a Gaussian parameter σ_0, and sample $(\mathbf{A}, \mathbf{A}_{\sigma_0}^{-1}) \leftarrow \mathsf{TrapGen}(n, m, q)$, $\mathbf{U} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{B} \xleftarrow{\$} \mathbb{Z}_q^{n \times md}$.
3. Output public parameters $\mathsf{pp} := (\mathbf{A}, \mathbf{B}, \mathbf{U})$ and master secret key $\mathsf{msk} := \mathbf{A}_{\sigma_0}^{-1}$.

$\mathsf{DMIPE.Derive}(\mathsf{pp}, \mathsf{msk}, \vec{V})$: Taking as input public parameters pp, a master secret key msk and a list of d-dimensional vectors $\vec{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_k\}$, perform:

1. For each vector $\mathbf{v}_i$, evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}(f_{\mathbf{v}_i}, \mathbf{B})$ and compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$.
2. Set $\mathbf{B}_{\vec{V}} := [\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$ and $\mathbf{A}_{\vec{V}} := [\mathbf{A}|\mathbf{B}_{\vec{V}}]$.
3. Compute trapdoor $\mathbf{A}_{\vec{V},\sigma_0}^{-1}$ for $\mathbf{A}_{\vec{V}}$ (via Item 3 of Lemma 7) and output $\mathsf{sk}_{\vec{V}} := \mathbf{A}_{\vec{V},\sigma_0}^{-1}$.

$\mathsf{DMIPE.Del}(\mathsf{pp}, \mathsf{sk}_{\vec{V}_1}, \mathbf{v}_{k+1})$: On input public parameters pp, a secret key $\mathsf{sk}_{\vec{V}_1} = \mathbf{A}_{\vec{V}_1,\sigma_0}^{-1}$ 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:

1. 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 $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$.
2. 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, $\mathbf{A}_{\vec{V}_1} := [\mathbf{A}|\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$
3. 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 7) and output $\mathsf{sk}_{\vec{V}_2} := \mathbf{A}^{-1}_{\vec{V}_2,\sigma_0}$.

DMIPE.Enc$(\mathsf{pp}, \mu, \mathbf{x})$**:** On input public parameters pp, a message vectors $\mu := (\mu_1, \cdot, \mu_m) \in \{0,1\}^m$ and an attribute vector $\mathbf{x} \in \mathbb{Z}_q^d$, do the following:

1. Sample $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{R} \xleftarrow{\$} \{-1,0,1\}^{m\times md}$ and $\mathbf{e}_{\mathsf{in}}, \mathbf{e}_{\mathsf{out}} \leftarrow \chi^m$.
2. Compute $\mathbf{c}_{\mathsf{in}} := \mathbf{s}^\top\mathbf{A} + \mathbf{e}_{\mathsf{in}}^\top \in \mathbb{Z}_q^m$, $\mathbf{c}_{\mathsf{mid}} := \mathbf{s}^\top(\mathbf{B} - \mathbf{x}\otimes\mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top\mathbf{R} \in \mathbb{Z}_q^{md}$, $\mathbf{c}_{\mathsf{out}} := \mathbf{s}^\top\mathbf{U} + \mathbf{e}_{\mathsf{out}}^\top + \mu\cdot\lceil q/2\rceil \in \mathbb{Z}_q^m$.
3. Output ciphertext $\mathsf{ct}_{\mathbf{x}} := (\mathbf{c}_{\mathsf{in}}, \mathbf{c}_{\mathsf{mid}}, \mathbf{c}_{\mathsf{out}})$.

DMIPE.Dec$(\mathsf{pp}, \mathsf{sk}_{\vec{V}}, \mathsf{ct}_{\mathbf{x}})$**:** On input public parameters 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:

1. 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 $\mathbf{B}_{\mathbf{v}_i} := \mathbf{B}\mathbf{H}_{\mathbf{v}_i}$.
2. Set $\mathbf{A}_{\vec{V}} := [\mathbf{A}|\mathbf{B}_{\mathbf{v}_1}|\cdots|\mathbf{B}_{\mathbf{v}_k}]$.
3. Compute $\mathbf{W} \leftarrow \mathbf{A}^{-1}_{\vec{V},\sigma_0}(\mathbf{U})$, i.e., $\mathbf{A}_{\vec{V}}\mathbf{W} = \mathbf{U} \pmod q$.
4. For $i \in [k]$, compute $\mathbf{c}_{\mathbf{v}_i} := \mathbf{c}_{\mathsf{mid}}\mathbf{H}_{\mathbf{v}_i}$, i.e, $\mathbf{c}_{\mathbf{v}_i} = \mathbf{s}^\top(\mathbf{B}_{\mathbf{v}_i} + \langle\mathbf{v}_i, \mathbf{x}\rangle\cdot\mathbf{G}) + \mathbf{e}_{\mathsf{in}}^\top\mathbf{R}\mathbf{H}_{\mathbf{v}_i}$.
5. Compute $\mu' := (\mu'_1, \cdots, \mu'_m) \leftarrow \mathbf{c}_{\mathsf{out}} - [\mathbf{c}_{\mathsf{in}}|\mathbf{c}_{\mathbf{v}_1}|\cdots|\mathbf{c}_{\mathbf{v}_k}]\mathbf{W}$.
6. 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. 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

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

Since we have

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

Here, the second and the third inequality are due to Lemma 3. The last inequality is due to $\|\mathbf{e}_{\mathsf{in}}^\top\|_{\max} \le B$ (as χ is B-bounded), $\|\mathbf{R}\|_{\max} \le 1$, $\|\mathbf{H}_{\mathbf{v}_i}\|_{\max} \le 1$ (Lemma 8) and $\|\mathbf{W}\|_{\max} \le 12\sigma_0$ (by Item 5 of Lemma 7). By choosing parameters such that $B + 12(mB + km^3B)\cdot\sigma_0 \le q/4$, the theorem follows. □

Setting Parameters. Parameters should be chosen as follows:

- First, choose λ to be a security parameter.
- For the hardness of $(n, 2m, q, \chi)$-DLWE (in Lemma 13): by Lemma 6, we choose $n = n(\lambda)$, ϵ, $q = q(n) \leq 2^n$, $m = \Theta(n \log q) = \mathsf{poly}(n)$, $\chi = \chi(n)$ such that χ is a (B, ϵ)-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 [6, Section 5.2.1].
- $m > (n+1)\log q + \omega(\log n)$ (For Lemma 11; due to Lemma 5).
- $\sigma_0 = \omega(n \log q \log n)$ (for $\mathsf{TrapGen}$; due to Item 1 of Lemma 7)
- $\sigma \geq m^2 d \cdot \omega(\sqrt{\log m})$ (for Hybrid 3 to work; due to Lemma 12).
- $B + 12(mB + km^3 B) \cdot \sigma_0 < q/4$. (for Correctness; due to Theorem 3).

We come up with the selective payload-hiding security of the proposed DMIPE.

Theorem 4 (Selective Payload-hiding Security). *Under the hardness of the $(n, 2m, q, \chi)$-*DLWE *assumption, the lattice-based* 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 $(n, 2m, q, \chi)$-*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 Hybrid i. We want to prove that $|\Pr[\mathcal{W}_0] - 1/2| = \mathsf{negl}(\lambda)$

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 $\mathbf{R}^* \in \{-1, 0, 1\}^{m \times md}$. Also, let $\mathbf{x}^*$ be the target attribute vector released by the adversary.

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

Hybrid 2 This hybrid is similar to Hybrid 1 except the way the challenger sets public parameters 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 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 7. Specifically,

1. For each vector $\mathbf{v}_i$, evaluate $\mathbf{H}_{\mathbf{v}_i} \leftarrow \mathsf{EvalF}^{\mathsf{IP}}(f_{\mathbf{v}_i}, \mathbf{AR}^*)$.
2. Compute $\mathbf{B}_{\mathbf{v}_i} := \mathbf{BH}_{\mathbf{v}_i} = \mathbf{AR}^*\mathbf{H}_{\mathbf{v}_i} + \langle \mathbf{v}_i, \mathbf{x}^* \rangle \cdot \mathbf{G}$.
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 $[\mathbf{A}|\mathbf{AR}^*\mathbf{H}_{\mathbf{v}_{i_0}} + \langle \mathbf{v}_{i_0}, \mathbf{x}^* \rangle \cdot \mathbf{G}]_\sigma^{-1}$ from $\mathbf{G}_{O(1)}^{-1}$ (Item 4 of Lemma 7),
   (b) compute $[\mathbf{A}_{\vec{V}}]_\sigma^{-1}$ from $[\mathbf{A}|\mathbf{AR}^*\mathbf{H}_{\mathbf{v}_{i_0}} + \langle \mathbf{v}_{i_0}, \mathbf{x}^* \rangle \cdot \mathbf{G}]_\sigma^{-1}$ (Item 3 of Lemma 7), and finally assign $\mathsf{sk}_{\vec{V}} \leftarrow [\mathbf{A}_{\vec{V}}]_\sigma^{-1}$.

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.

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

Lemma 10. *In the view of the adversary $\mathcal{A}$,* Hybrid 1 *and* Hybrid 0 *are perfectly the same; i.e.,* $\Pr[\mathcal{W}_1] = \Pr[\mathcal{W}_0]$.

Proof. 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 noticed. □

Lemma 11. *In the view of the adversary $\mathcal{A}$,* Hybrid 2 *and* Hybrid 1 *are indistinguishable, i.e.,* $|\Pr[\mathcal{W}_1] - \Pr[\mathcal{W}_2]| = \mathsf{negl}(\lambda)$.

Proof. This is simply due to the leftover hash lemma (Lemma 5). □

Lemma 12. *In the view of the adversary $\mathcal{A}$,* Hybrid 3 *and* Hybrid 2 *are indistinguishable, i.e.,* $|\Pr[\mathcal{W}_2] - \Pr[\mathcal{W}_3]| = \mathsf{negl}(\lambda)$.

Proof. This is simply due to (i) the pseudo-randomness of TrapGen (see Item 1 of Lemma 7) 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 σ in Step 4 of Hybrid 3. Namely, we should choose

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

□

Lemma 13. *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* $(n, 2m, q, \chi)$-DLWE *problem.*

Proof. 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 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 μ_0^* and μ_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.

□

Now, we have $|\Pr[\mathcal{W}_0]| \le |\Pr[\mathcal{W}_0] - \Pr[\mathcal{W}_1]| + |\Pr[\mathcal{W}_1] - \Pr[\mathcal{W}_2]| + |\Pr[\mathcal{W}_2] - \Pr[\mathcal{W}_3]| + |\Pr[\mathcal{W}_3] - \Pr[\mathcal{W}_4]| = \mathsf{negl}(\lambda)$. □

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 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.

$\mathsf{DMIPE.Setup}(1^\lambda, \mathsf{sp})$: Run $(\mathsf{se.pp}, \mathsf{se.msk}) \leftarrow \mathsf{SE.Setup}(1^\lambda, \mathsf{sp})$ and then output $\mathsf{pp} := \mathsf{se.pp}$, $\mathsf{msk} := \mathsf{se.msk}$.

$\mathsf{DMIPE.Derive}(\mathsf{pp}, \mathbf{T}, \mathsf{msk}, \vec{V})$: Run $V \leftarrow \mathsf{OVS}(\vec{V})$ and $\mathsf{se.sk}_V \leftarrow \mathsf{SE.Derive}(\mathsf{pp}, \mathsf{msk}, V)$ 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 $\mathsf{SE.Dec}(\mathsf{pp}, \mathsf{ct}_{\mathbf{x}}, \mathsf{sk}_V)$.

The correctness of DMIPE is straightforward from that of SE. The security of DMIPE folllows from that of SE can be done in a similar way as in the proof of Theorem 2. Then we omit it.

7 Allow-/Deny-list Encryption from Spatial Encryption

We generalize IBE [39], HIBE [25], PE [28], DFPE [22], FuPE [21] in a family called Allow-/Deny-list Encryption (ADE). 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 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.

The formal syntax and the security notions for ADE will be given next. We introduce three versions of ADE: (i) standard ADE (sADE); (ii) k-threshold ADE (k-tADE) and (iii) inclusive ADE (iADE depending on the correctness requirements. After that, we present encodings that help to construct sADE and iADE from SE.

7.1 Framework of ADE

Let λ 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 ADE=(ADE.Gen, ADE.Enc, ADE.Npun, ADE.Ppun, ADE.Dec):

$(\mathsf{pp}, \mathsf{sk}_{\emptyset}^{\emptyset}) \leftarrow$ **ADE.Gen**$(1^{\lambda}, 1^{a}, 1^{d})$**:** On input (a security parameter λ 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 pp and a (not punctured) initial secret key $\mathsf{sk}_{\emptyset}^{\emptyset}$.

$\mathsf{sk}_{DL'}^{AL'_1 \cup AL'_2} \leftarrow$ **ADE.Ppun**$(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'_1}, AL'_2, k)$**:** [6] On input a tuple of (public parameters 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 $\mathsf{sk}_{DN'}^{A'L \cup \{\mathsf{pt}\}}$.

$\mathsf{sk}_{DL'_1 \cup DL'_2}^{AL'} \leftarrow$ **ADE.Npun**$(\mathsf{pp}, \mathsf{sk}_{DL'_1}^{AL'}, DL'_2)$**:** On input a tuple of (public parameters pp; a previously punctured key $\mathsf{sk}_{DL'}^{AL'}$ 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 $\mathsf{sk}_{DL'_1 \cup DL'_2}^{AL'}$.

[6] Here, note that k is only used in the k-tADE variant.

$\mathbf{ct}_{DL}^{AL} \leftarrow$ **ADE.Enc**$(\mathsf{pp}, \mu, AL, DL)$**:** On input a tuple of (public parameters pp; a plaintext μ; a set of positive tags AL; a set of negative tags DL), the PPT algorithm returns a ciphertext ct_{DL}^{AL}.

$\mu/\perp \leftarrow$ **ADE.Dec**$(\mathsf{pp}, \mathbf{sk}_{DL'}^{AL'}, \mathbf{ct}_{DL}^{AL})$**: :** On input a tuple of (public parameters pp; a secret key $\mathsf{sk}_{DL'}^{AL'}$ associated with $AL' \subseteq \mathcal{T}^{(+)}$ and $DL' \subseteq \mathcal{T}^{(-)}$; a ciphertext ct_{DL}^{AL} associated with $AL \subseteq \mathcal{T}^{(+)}$ and $DL \subseteq \mathcal{T}^{(-)}$), the DPT algorithm outputs either a plaintext μ if decryption succeeds or $\perp$ otherwise.

Correctness and ADE Variants. Consider all $\lambda, a, d \in \mathbb{N}$, $\mu \in \mathcal{M}$, $\emptyset \subset AL, AL' \subseteq \mathcal{T}^{(+)}, \emptyset \subset DL, DL' \subseteq \mathcal{T}^{(-)}$, $(\mathsf{pp}, \mathsf{sk}_{\emptyset}^{\emptyset}) \leftarrow \mathsf{ADE.Gen}(1^{\lambda}, 1^{a}, 1^{d})$, $\mathsf{ct}_{DL}^{AL} \leftarrow \mathsf{ADE.Enc}\ (\mathsf{pp}, \mu, AL, DL)$, and any punctured key $\mathsf{sk}_{DL'}^{AL'}$ 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., $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}_{\emptyset}^{\emptyset}, \mathsf{ct}_{DL}^{AL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. 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 $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'}, \mathsf{ct}_{DL}^{AL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. Otherwise, $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'}, \mathsf{ct}_{DL}^{AL}) = \mu] \leq \mathsf{negl}(\lambda)$.
2. **Inclusive ADE (iADE).** If $((AL' \subseteq AL) \wedge (DL \cap DL' = \emptyset))$ then $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'}, \mathsf{ct}_{DL}^{AL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. Otherwise, $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'}, \mathsf{ct}_{DL}^{AL}) = \mu] \leq \mathsf{negl}(\lambda)$.
3. **k-threshold ADE (k-tADE).** If $((|AL \cap AL'| \geq k) \wedge (DL \cap DL' = \emptyset))$, then $\Pr[\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'}, \mathsf{ct}_{DL}^{AL}) = \mu] \geq 1 - \mathsf{negl}(\lambda)$. Otherwise, $\Pr[\mathsf{ADE.Dec}\ (\mathsf{pp}, \mathsf{sk}_{DL'}^{AL'}, \mathsf{ct}_{DL}^{AL}) = \mu] \leq \mathsf{negl}(\lambda)$.

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 (see Definition 4 and Figure 4).

Definition 4. *ADE is selective payload-hiding secure if the advantage of the adversary playing in the* $\mathsf{ADE}_{\mathsf{payload},\mathcal{A}}^{\mathsf{sel},\mathsf{ATK}}$ *game (in Figure 4) is negligible or*

$$\mathsf{Adv}_{\mathsf{ADE},\mathcal{A},\mathsf{sel}}^{\mathsf{payload},\mathsf{ATK}} := |\Pr[b' = b] - 1/2| = \mathsf{negl}(\lambda).$$

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 $(AL'_1, DL'_1), (AL'_2, DL'_2) \in \mathcal{T}^{(+)} \times \mathcal{T}^{(-)}$, 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:

$\mathsf{ADE}^{\mathsf{sel},\mathsf{ATK}}_{\mathsf{payload},\mathcal{A}}$ **Game** ($\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$) :

- $(AL^*, DL^*) \leftarrow \mathcal{A}(1^\lambda, 1^a, 1^d)$;
- $(\mathsf{pp}, \mathsf{sk}_\emptyset^\emptyset) \leftarrow \mathsf{ADE.Gen}(1^\lambda, 1^a, 1^d)$, $AL' \leftarrow \emptyset$, $DL' \leftarrow \emptyset$, $\mathsf{ADKList} \leftarrow \emptyset$;
  // $\mathsf{ADKList}$ is to store $(AL', DL', \mathsf{sk}_{DL'}^{AL'})$
- $(\mu_0^*, \mu_1^*) \leftarrow \mathcal{A}^{\mathsf{Punc}(\cdot,\cdot),\mathsf{DQ}(\cdot,\cdot)}(\mathsf{pp})$;
- $b \xleftarrow{\$} \{0,1\}$, $\mathsf{ct}_{DL^*}^{AL^*} \leftarrow \mathsf{ADE.Enc}(\mathsf{pp}, \mu_b^*, AL^*, DL^*)$;
- $b' \leftarrow \mathcal{A}^{\mathsf{Punc}(\cdot,\cdot),\mathsf{DQ}(\cdot)}(\mathsf{pp}, \mathsf{ct}_{DL^*}^{AL^*})$. // **Restrictions:** Not allowed $\mathsf{DQ}(AL', DL', \mathsf{ct}_{DL^*}^{AL^*})$ with $(AL', DL') \in \mathsf{SUCC}(AL^*, DL^*)$.

Queried Oracles:

- $\mathsf{Punc}((AL'_1, DL'_1), (AL'_2, DL'_2))$ (It is only allowed if $(AL'_1 \cup AL'_2, DL'_1 \cup DL'_2) \notin \mathsf{SUCC}(AL^*, DL^*)$): Take $\mathsf{sk}_{DL'_2}^{AL'_1}$ from $(AL'_1, DL'_1, \mathsf{sk}_{DL'_1}^{AL'_1}) \in \mathsf{ADKList}$ (if exists there). Otherwise, run $\mathsf{ADE.Ppun}$ and $\mathsf{ADE.Npun}$ in any order using $\mathsf{sk}_\emptyset^\emptyset$ to get $\mathsf{sk}_{DL'_1}^{AL'_1}$. Finally, run $\mathsf{ADE.Ppun}$ and $\mathsf{ADE.Npun}$ in any order using $\mathsf{sk}_{DL'_2}^{AL'_1}$ to output $\mathsf{sk}_{DL'_1 \cup DL'_2}^{AL'_1 \cup AL'_2}$. Update $\mathsf{ADKList} \leftarrow \mathsf{ADKList} \cup \{(AL'_1 \cup AL'_2, DL'_1 \cup DL'_2, \mathsf{sk}_{DL'_1 \cup DL'_2}^{AL'_1 \cup AL'_2})\}$.
- $\mathsf{DQ}(AL', DL', \mathsf{ct}_{DL}^{AL})$ (allowed only if ATK=CCA): Take $\mathsf{sk}_{DL'}^{AL'}$ from $(AL', DL', \mathsf{sk}_{DL'}^{AL'}) \in \mathsf{ADKList}$ (if exists there). Otherwise, run $\mathsf{ADE.Ppun}$ and $\mathsf{ADE.Npun}$ in any order using $\mathsf{sk}_\emptyset^\emptyset$ to get $\mathsf{sk}_{DL'}^{AL'}$. Finally, return the output of $\mathsf{ADE.Dec}(\mathsf{pp}, \mathsf{ct}_{DL}^{AL}, \mathsf{sk}_{DL'}^{AL'})$. Update $\mathsf{ADKList} \leftarrow \mathsf{ADKList} \cup \{(AL', DL', \mathsf{sk}_{DL'}^{AL'})\}$.

Define $\mathsf{SUCC}(AL^*, DL^*)$ **for ADE Variants:**

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

Fig. 4: Selective security for the ADE variants

$W_{\mathsf{key}} \leftarrow$ **EncodeInKey**(AL', DL')**.** Do the following:

1. Associate the allow list $AL' = \{p_1, \cdots, p_k\}$ with a space beginning with $(p_1, \cdots, p_k)^\top$, namely $W_{AL'} := \{(p_1, \cdots, p_k, x_{k+1}, \cdots, x_a)^\top : x_i \in \mathbb{Z}_q\} \subseteq \mathbb{Z}_q^a$.
   Obviously, it is easy to see that if $AL'_1 \subseteq AL'_2$ then $W_{AL'_2} \sqsubseteq W_{AL'_1}$.
2. 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 $W_{DL''_2} \sqsubseteq W_{DL''_1}$.
3. Output the subspace W_{key} which is the direct product of $W_{AL'}$ and $W_{DL'}$:
$$W_{\mathsf{key}} := W_{AL'} \times W_{DL'}.$$

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

$\mathbf{v}_{\mathsf{ct}} \leftarrow$ **EncodeInCipher**(AL, DL)**.** Do the following steps:

1. For $AL = \{p_1, \cdots, p_k\}$ associate with vector $\mathbf{x}_{AL} := (p_1, \cdots, p_k, 0, \cdots, 0) \in \mathbb{Z}_q^a$.
   Clearly, if $AL' \subseteq AL$ then $\mathbf{x}_{AL} \in W_{AL'}$.

2. For a list DL, encode it as $\mathbf{x}_{DL} := \sum_{x \in DL} \mathbf{v}_x \in \mathbb{Z}_q^{2d}$, 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., $DL \nsubseteq DL''$).
3. Output vector $\mathbf{x}_{\mathsf{ct}} := (\mathbf{x}_{AL}, \mathbf{x}_{DL}) \in \mathbb{Z}_q^{a+2d}$.

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 that 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 Delegatable Multiple Inner Product Encryption (DMIPE). We show that DMIPE is sufficient for us to build an efficient lattice-based Spatial Encryption (SE). The lattice-based SE is more efficient than some previous lattice-based ones, which follow the generic SE construction from the Hierarchical Inner Product Encryption. In fact, DMIPE and SE are equivalent in the sense that there are "security notions-preserving" conversions between them.

Although, our lattice-based DMIPE is proved to be selectively payload-hiding secure in the standard model. However, it seems that the construction can enjoy the selectively weak attribute-hiding security. A possible technical idea might be from Agrawal et al. [4]. We leave this for a future work.

Moreover, an adaptively secure DMIPE construction in the lattice setting is worthwhile to pursuit in the future. Recall that, such a construction for IPE has been done for IPE by [30]. 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 the Hamburg's thesis [29, Page 51] can help. However, the Douby Spatial Encryption (DSE) or another SE variant rather than original SE (as defined in this paper) might be needed.

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., Boneh, D., Boyen, X.: Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **6223 LNCS**, 98–115 (2010). https://doi.org/10.1007/978-3-642-14623-7_6
4. 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

5. 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
6. Alkim, E., Bos, J.W., Ducas, L., Others: Frodo{KEM}: Learning with Errors Key Encapsulation Algorithm Specifications And Supporting Documentation, version 30 September, 2020 (2020)
7. Alperin-Sheriff, J., Peikert, C.: Faster Bootstrapping with Polynomial Error. In: Garay, J.A., Gennaro, R. (eds.) Advances in Cryptology – CRYPTO 2014. pp. 297–314. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
8. Alwen, J., Peikert, C.: Generating Shorter Bases for Hard Random Lattices. In: 26th International Symposium on Theoretical Aspects of Computer Science, {STACS} 2009, February 26-28, 2009, Freiburg, Germany, Proceedings. pp. 75–86 (2009). https://doi.org/10.4230/LIPIcs.STACS.2009.1832
9. 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
10. 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
11. Boneh, D., Kim, S., Montgomery, H.: Private Puncturable PRFs from Standard Lattice Assumptions. In: Coron, J.S., Nielsen, J.B. (eds.) Advances in Cryptology – EUROCRYPT 2017. pp. 415–445. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_15
12. 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
13. Brakerski, Z., Langlois, A., Peikert, C., Regev, O., Stehlé, D.: Classical Hardness of Learning with Errors. In: Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing. pp. 575–584. STOC '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2488608.2488680
14. Brakerski, Z., Tsabary, R., Vaikuntanathan, V., Wee, H.: Private Constrained PRFs (and More) from LWE. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **10677 LNCS**(H2020 639554), 264–302 (2017). https://doi.org/10.1007/978-3-319-70500-2_10
15. 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
16. 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
17. 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

18. 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
19. 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
20. Cohen, H.: A Course in Computational Algebraic Number Theory. No. Graduate texts in mathematics, 138, Springer, Berlin (1996)
21. 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
22. 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)
23. 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
24. 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)
25. 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
26. Gorbunov, S., Vaikuntanathan, V., Wee, H.: Attribute-Based Encryption for Circuits. J. ACM **62**(6), 45:1—-45:33 (dec 2015). https://doi.org/10.1145/2824233
27. 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
28. Green, M.D., Miers, I.: Forward Secure Asynchronous Messaging from Puncturable Encryption. In: 2015 IEEE Symposium on Security and Privacy. pp. 305–320 (may 2015). https://doi.org/10.1109/SP.2015.26
29. Hamburg, M.: Spatial Encryption. PhD. Thesis (July) (2011)
30. 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
31. 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
32. 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
33. 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)
34. Okamoto, T., Takashima, K.: Homomorphic Encryption and Signatures from Vector Decomposition. In: Galbraith, S.D., Paterson, K.G. (eds.) Pairing-Based Cryp-

tography – Pairing 2008. pp. 57–74. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85538-5_4
35. 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
36. Peikert, C.: Bonsai Trees (or, Arboriculture in Lattice-Based Cryptography). Cryptology ePrint Archive, Report 2009/359 (2009)
37. Peikert, C., Shiehian, S.: Privately Constraining and Programming PRFs, the LWE Way. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **10770 1076**, 675–701 (2018). https://doi.org/10.1007/978-3-319-76581-5_23
38. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM **56**(6), 84–93 (sep 2009). https://doi.org/10.1145/1568318.1568324
39. 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
40. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134 (nov 2002). https://doi.org/10.1109/sfcs.1994.365700
41. 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
42. 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
43. Wei, J., Chen, X., Wang, J., Hu, X., Ma, J.: Forward-Secure Puncturable Identity-Based Encryption for Securing Cloud Emails. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). vol. 11736 LNCS, pp. 134–150 (2019). https://doi.org/10.1007/978-3-030-29962-0_7
44. Xagawa, K.: Improved (Hierarchical) Inner-Product Encryption from Lattices. Full version of the paper appeared at PKC'13 pp. 235–252 (2015)
45. 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