Fully Secure (Doubly-)Spatial Encryption under Simpler Assumptions

Cheng Chen, Zhenfeng Zhang, and Dengguo Feng

State Key Laboratory of Information Security, Institute of Software, Chinese Academy of Sciences, Beijing, China
{chencheng, zfzhang, feng}@is.iscas.ac.cn

Abstract. Spatial encryption was first proposed by Boneh and Hamburg in 2008. It is one implementation of the generalized identity-based encryption schemes and many systems with a variety of properties can be derived from it. Recently, Hamburg improved the notion by presenting a variant called doubly-spatial encryption. The doubly-spatial encryption is more powerful and expressive. More useful cryptography systems can be builded from it, such as attribute-based encryption, etc. However, most presented spatial encryption schemes are proven to be selectively secure. Only a few spatial encryption schemes achieve adaptive security, but not under standard assumptions. And no fully secure doubly-spatial encryption scheme has been presented before.

In this paper, we primarily focus on the adaptive security of (doubly-)spatial encryption. A spatial encryption scheme and a doubly-spatial encryption scheme have been proposed. Then we apply the dual system methodology proposed by Waters in the security proof. Both of the schemes can be proven adaptively secure under standard assumptions, the decisional linear (DLIN) assumption and the decisional bilinear Diffe-Hellman (DBDH) assumption, over prime order groups in the standard model. To the best of our knowledge, our second scheme is the first fully secure construction of doubly-spatial encryption. [1]

keywords: spatial encryption, adaptive security, dual system encryption

1 Introduction

Identity-based cryptography, proposed by Shamir in 1984 [15]: it allows any string to be used as a public key. In an identity-Based Encryption (IBE) scheme: an authority that holds a master secret key can take any arbitrary identity string and extract a secret key corresponding to this identity. A message is encrypted under the recipient's identity and only the user with the matching identity can successfully decrypt the message.

In 2008, Boneh and Hamburg [2] proposed a general framework for constructing identity-based crypto systems, which they called Generalized Identity-Based Encryption (GIBE). In the GIBE, one secret key $SK_{\mathfrak{r}}$ is associated with a role $\mathfrak{r}$, which belongs to an allowable roles set $\mathfrak{R}$. Secret keys for certain roles can be delegated to create keys for other roles. A role $\mathfrak{r}$ can delegate to another role $\mathfrak{r}'$, and defines $\mathfrak{r} \succeq \mathfrak{r}'$, if a key for $\mathfrak{r}$ can be used to efficiently create a key for $\mathfrak{r}'$. Because this relation is transitive and antisymmetric, $\succeq$ defines a partial order on $\mathfrak{R}$. Let $\mathfrak{P}$ be a set of allowable policies. A user should choose a policy $\mathfrak{p} \in \mathfrak{P}$ to encrypt a message, and outputs $CT_{\mathfrak{p}}$. The chosen policy governs which users will be able to decrypt the message using the keys corresponding to their roles. If $SK_{\mathfrak{r}}$ can decrypt $CT_{\mathfrak{p}}$, defines $\mathfrak{r} \succeq \mathfrak{p}$. Further more, the systems must have a most powerful role $\top \in \mathfrak{R}$. The master secret key in GIBE $SK_{\top}$ has a role $\top$. For all policies $\mathfrak{p}$ and roles $\mathfrak{r}$, we have $\top \succeq \mathfrak{r}$ and $\top \succeq \mathfrak{p}$. So the master secret key can be seen as a secret key that has highest power in the system. The roles and policies can be efficiently encoded, and that the $\preceq$ relation can be determined efficiently.

[2] also gave an important instance of the GIBE framework: spatial encryption. In a spatial encryption scheme, policies are points in $\mathbb{Z}_p^n$ and roles are affine subspaces of $\mathbb{Z}_p^n$. The delegation

relation $\succeq$ on roles is defined by subspace inclusion: role $\rho_1 \succeq \rho_2$ if ρ_1's affine subspace contains ρ_2's subspace. The ciphertext size is always constant. With the property, the spatial encryption scheme can be used to construct a host of efficient IBE-like schemes: multicast IBE, broadcast hierarchical IBE, predicate encryption, multiple authorities IBE and so on.

In [8], Hamburg first proposed the notion of doubly-spatial encryption. Doubly-spatial encryption is more expressive than spatial encryption. In the doubly-spatial encryption scheme, the policies are affine subspaces in $\mathbb{Z}_p^n$ instead of vectors. And the secret key can decrypt the ciphertext if its affine subspace intersects with the affine subspace of the ciphertext. As mentioned in [8], many useful crypto systems can be implemented by doubly-spatial encryption scheme, such as attribute-based encryption [7,14], threshold-based encryption [3,5], all-but-one signatures [8], etc. Spatial encryption can be embedded in doubly-spatial encryption. But the construction of doubly-spatial encryption is not so efficient as spatial encryption, since the length of the ciphertext is not constant. The first construction of doubly-spatial encryption [8] is proven to be selectively secure under some unnatural assumptions.

Related Work. Boneh and Hamburg [2] proposed the first selectively secure spatial encryption under Bilinear Decision Diffie-Hellman Exponent (BDHE) assumption. Subsequently, Zhou and Cao [17] provided a variant of Boneh-Hamburg scheme under a weaker assumption, decisional bilinear Diffe-Hellman (DBDH) assumption, but the ciphertext size is not constant. The first fully secure spatial encryption scheme was proposed by [12]. The construction [12] was based on the three composite order bilinear groups and proven fully secure under three non-standard assumptions over composite order bilinear groups. Recently, Hamburg [8] proposed an adaptively secure scheme based on some static assumptions over prime order groups, but the assumptions are still non-standard. [8] also proposed the first doubly-spatial encryption with selective security. Attrapadung and Libert [1] proposed a constant-size ciphertext inner product encryption with adaptive security. The scheme can be regarded as a special case of spatial encryption, in which the vector of a secret key can be seen as the orthogonal space of the vector in the spatial encryption. Till now, no spatial or doubly-spatial encryption scheme can been proven fully secure under some natural assumptions. As [8] said, how to construct fully secure spatial and doubly-spatial encryption schemes using natural assumptions is still a problem that needs to solve.

Waters [16] introduced the dual system encryption to overcome the limitations of partitioning. In a dual encryption system, keys and ciphertexts can take on one of two forms: normal and semi-functional. A normal key can decrypt both normal and semi-functional ciphertexts, while a semi-functional key can only decrypt normal ciphertexts. The semi-functional keys and ciphertexts are not used in the real system, only in the proof of security. And later, dual system encryption used in [9–11, 13] to obtain adaptive security for IBE, HIBE, and ABE systems. Since prime order groups do not have the good functionalities and appealing features as composite order groups, only a few schemes [10,13] can be realizing in the prime order setting. Though [6] proposed techniques translation from composite order schemes to prime order one, the translations are insufficient and inefficient and how to efficiently achieve fully secure in prime order setting remains an interesting issue.

Our Contributions. The main drawbacks of the previous works are that the schemes are only selectively secure or they achieve adaptively secure using some complex assumptions. In this paper, we construct a fully secure spatial encryption scheme and a fully secure doubly-spatial encryption scheme. And we use dual system encryption technique introduced by Waters [16] in the proof. In contrast with previous spatial encryption works, the security of our scheme depends on neither some non-standard assumptions [8] nor the assumptions over composite order pairing groups [12], but two standard assumptions: DLIN and DBDH, in the standard model. Our spatial encryption scheme has constant-size ciphertext, which coincides with the original intention of spatial encryption. Our doubly-spatial encryption scheme can been seen as an extension of the first scheme. Its security is also reduced to DLIN and DBDH assumptions in the standard model. To remark, our doubly-spatial encryption scheme is the first fully secure construction. This paper solves the problem brought forward by Hamburg in [8].

Our schemes are based on Waters' tag-based IBE [16]. In the constructions, we extend the "two-equation revocation" technique of [10] to "n-equation revocation". We create each ciphertext with a uniformly distributed tag and each secret key with a uniformly distributed tag, too. The decryption algorithm will not work if the tag of the secret key and the tag of ciphertext has some relations. While in the actual simulation from normal secret key to semi-functional secret key, the tags created by simulator are linear dependent. The simulator can create the semi-functional secret keys of all affine spaces in the vector space. All the semi-functional secret keys can not decrypt the challenged ciphertext, even if the affine spaces of the semi-functional secret keys contain the challenged vector due to the setting of tags. With the linear relationship of the tags, the simulation can process successfully. But this relationship is information theoretically hidden to the adversary.

2 Preliminaries

2.1 Bilinear Groups

We present a few facts related to groups with efficiently computable bilinear maps. $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that $e(g, g) \neq 1$ for g and for any u, $v \in \mathbb{Z}_p$, it holds that $e(g^u, g^v) = e(g, g)^{uv}$. We say that $\mathcal{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^u, g^v) = e(g, g)^{uv} = e(g^v, g^u)$.

2.2 Complexity Assumptions

We define the **Decisional Bilinear Diffie-Hellman** (DBDH) and **Decisional Linear** (DLIN) assumptions as follows.

Definition 1. *Let $\mathbb{G}$ be a bilinear group of prime order p as defined above. Choose a random generator $g \in \mathbb{G}$ and exponents $c_1, c_2, c_3 \in \mathbb{Z}_p$. An algorithm $\mathcal{B}$ that outputs $\mu \in \{0, 1\}$ has advantage ϵ in solving the DBDH problem if*

$$|Pr[\mathcal{B}(g, g^{c_1}, g^{c_2}, g^{c_3}, T = e(g, g)^{c_1 c_2 c_3}) = 0] - Pr[\mathcal{B}(g, g^{c_1}, g^{c_2}, g^{c_3}, T = R) = 0]| \geqslant \epsilon$$

where R is the random choice of $\mathbb{G}_T$. We say that the decision DBDH assumption holds in $\mathbb{G}$ if no polynomial time algorithm has non-negligible advantage in solving the DBDH problem.

Definition 2. *Choose random generators $g, f, \nu \in \mathbb{G}$ and exponents $c_1, c_2 \in \mathbb{Z}_p$. An algorithm $\mathcal{B}$ that outputs $\mu \in \{0, 1\}$ has advantage ϵ in solving the DLIN problem if*

$$|Pr[\mathcal{B}(g, f, \nu, g^{c_1}, f^{c_2}, T = \nu^{c_1+c_2}) = 0] - Pr[\mathcal{B}(g, f, \nu, g^{c_1}, f^{c_2}, T = R) = 0]| \geqslant \epsilon$$

where R is the random choice of $\mathbb{G}$. We say that the decision linear assumption holds in $\mathbb{G}$ if no polynomial time algorithm has non-negligible advantage in solving the DLIN problem.

2.3 Affine Spaces

Let p be a prime number, and $\mathbb{Z}_p$ be the field of integers modulo p. For some positive integer n, let $\mathbb{Z}_p^n$ denote an n-dimensional vector space over the field $\mathbb{Z}_p$. We use boldface to denote the vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_n) \in \mathbb{Z}_p^n$. The inner product is defined by $<\boldsymbol{a}, \boldsymbol{b}>= \sum_{i=1}^{n} a_i \cdot b_i$. For any vector $\boldsymbol{x} \in \mathbb{Z}_p^n$ and any matrix $M \in \mathbb{Z}_p^{n\times n}$ (w.l.o.g., we consider M as a phalanx in this paper), we define the affine subspace $S(M, \boldsymbol{x}) \subseteq \mathbb{Z}_p^n$ by

$$S(M, \boldsymbol{x}) := \{\boldsymbol{x} + M^\top \cdot \boldsymbol{y} | \boldsymbol{y} \in \mathbb{Z}_p^n\}$$

If these elements $M^\top \cdot \boldsymbol{y}$ are all unique, we have $S(M, \boldsymbol{x}) = \mathbb{Z}_p^n$ and we say that $S(M, \boldsymbol{x})$ is an n-dimensional affine subspace. It is a basic theorem from linear algebra that the dimension of an affine subspace $S(M, \boldsymbol{x})$ is the rank of M.

If $S(M', \boldsymbol{x}') \subseteq S(M, \boldsymbol{x}) \subseteq \mathbb{Z}_p^n$, we must have $M' = M \cdot T$ and $\boldsymbol{x}' = \boldsymbol{x} + M^\top \cdot \boldsymbol{y}$ for some (efficiently computable) matrix $T \in \mathbb{Z}_p^{n\times n}$ and $\boldsymbol{y} \in \mathbb{Z}_p^n$.

3 (Doubly-)Spatial Encryption

Below, we give the definition of (doubly-)spatial encryption and its security model.

3.1 Algorithms of (Doubly-)Spatial Encryption

A (doubly-)spatial encryption system is a quite expressive GIBE contribution that it can be embed many other GIBEs inside. The roles in the spatial encryption are all the affine subspaces of $\mathbb{Z}_p^n$. We say that $\mathfrak{r}_1 \preceq \mathfrak{r}_2$ if and only if $\mathfrak{r}_1 \subseteq \mathfrak{r}_2$. The policies for spatial encryption are vectors of $\mathbb{Z}_p^n$, and we say that $\mathfrak{p} \preceq \mathfrak{r}$ if and only if $\mathfrak{p} \in \mathfrak{r}$. For the doubly-spatial encryption, the policies are the affine subspaces in the vector space $\mathbb{Z}_p^n$. $\mathfrak{p} \preceq \mathfrak{r}$ if and only if $\mathfrak{p} \cap \mathfrak{r} \neq \varnothing$. The construction for spatial encryption will emerge as a special case of the construction for doubly-spatial encryption. A (doubly-)spatial encryption scheme consists of four polynomial time algorithms described as follows:

Setup(λ, n): The algorithm takes as input a security parameter λ and a space dimension n. It returns public parameters PP, an n-dimension affine space V and a master secret key $K_\top$. The master key $K_\top$ can be seen the secret key of the affine space K_V.

Delegate(PP, V_1, K_{V_1}, V_2): The algorithm takes as input the secret key K_{V_1} for the affine vector space V_1 and outputs the secret key K_{V_2} for V_2, where $V_1 \subseteq V_2$. We require that the distribution of the private keys produced by this algorithm should be independent of the path taken. That is, all keys for a given vector space come from the same distribution, no matter how they were delegated.

Encrypt$(PP, \boldsymbol{x}/W, m)$: The spatial encryption algorithm encrypts a message m under a vector $\boldsymbol{x}$ and outputs a ciphertext $CT_{\boldsymbol{x}}$. For the doubly-spatial case, the message m is encrypted under an affine subspace W, and the algorithm outputs a ciphertext CT_W.

Decrypt$(PP, CT_{\boldsymbol{x}}/CT_W, K_V, \boldsymbol{x}/W)$: The spatial decryption algorithm takes as input the secret key K_V to decrypt the ciphertext $CT_{\boldsymbol{x}}$. Decryption succeeds if $\boldsymbol{x} \in V$, and it outputs the plaintext m. For the doubly-spatial case, decryption succeeds if $W \cap V \neq \varnothing$, and it outputs the plaintext m.

3.2 Adaptive Security of (Doubly-)Spatial Encryption

We define the adaptive security for the (doubly-)spatial encryption system under chosen plaintext attacks (CPA) adversary. We now present the following game between an adversary and a challenger.

Setup(λ, n): The challenger runs the algorithm **Setup**(λ, n) and sends public parameters PP to the adversary.

Phase I: The adversary makes delegation queries of V_i to the challenger, who runs the delegation algorithm **Delegate**$(PP, \top, K_\top, V_i)$ and returns K_{V_i}.

Challenge: The adversary submits two messages m_0, m_1 and a vector (or an affine subspace) $\boldsymbol{x}/W$ for challenge. We require that the adversary has not been given a decryption key whose affine subspace contains the challenged vector (intersects with the challenged affine subspace for the doubly-spatial encryption case), that is, $\boldsymbol{x} \notin V_i$ $(W \cap V_i = \varnothing)$ for all delegation queries of V_i in Phase I. The challenger chooses a random $\mu \in \{0, 1\}$, runs the algorithm **Encrypt**$(PP, \boldsymbol{x}/W, m_\mu)$, and returns the resulting challenge ciphertext CT^* to the adversary.

Phase II: The second query phase is exactly like the first one, except that the adversary may not issue delegation queries for affine subspace that contains $\boldsymbol{x}$ (intersects with W).

Guess: The adversary outputs a guess $\mu' \in \{0, 1\}$ and wins if $\mu' = \mu$.

We define the advantage of the adversary $\mathcal{A}$ in attacking a (doubly-)spatial encryption scheme Π as $Adv^{SE}_{\mathcal{A},\Pi} = |Pr[\mu' = \mu] - 1/2|$.

Note that, we only define the delegation of secret keys from the master key in stead of other secret keys. Since the distribution of the secret keys are independent from the path of delegation taken, we can use the delegation of the master secret key to simulate all the other secret keys' delegations.

Definition 3. *A (doubly-)spatial encryption scheme Π is adaptively secure if no PPT adversaries have at most non-negligible advantage in winning the above game.*

4 Adaptively Secure Spatial Encryption under Simple Assumptions

In this section, we propose an adaptively secure spatial encryption based on [16] tag-based IBE. We attach a tag value $tag_c \in \mathbb{Z}_p$ to each ciphertext and a tag which is a vector $(tag_V, \boldsymbol{tag}_V) \in \mathbb{Z}_p^{n+1}$ to each secret key for space V. The decryption algorithm will only work if the tag of the decryption key and the tag of ciphertext have the relation $< M^\top \cdot \boldsymbol{y}, \boldsymbol{tag}_V > + tag_V - tag_c \neq 0$.

In addition, the delegation algorithm should guarantee the distribution of secret keys independent of the delegation path. However, the HIBE scheme in [16] dose not guarantee this property. Because the tag of the new secret key coincides with the existing secret key in delegation algorithm. While in the key generation algorithm, the tag of the secret key is randomly chosen. In order to avoid that, we only uniformly choose the tag of the master secret key, and let delegation algorithm compute the tag of the new key in some way instead of randomly choosing it. In the mean while, re-randomization is also required to uniform the distribution of the new secret keys.

Setup(λ, n): The setup algorithm takes input a security parameter λ. It first chooses bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^\lambda$ and an n-dimensional affine space $V = \mathbb{Z}_p^n$. Next, it randomly chooses generators $g, v, v_1, v_2 \in \mathbb{G}$, $\alpha, \alpha_0, \alpha_1, \ldots, \alpha_n, a_1, a_2, b, r_1, r_2, z_1, z_2, tag_V \in \mathbb{Z}_p$, and $\boldsymbol{tag}_V, \boldsymbol{x_0} \in \mathbb{Z}_p^n$. Let $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. It publishes the public parameters PP as the group description $\mathbb{G}$ along with:

$$g, g^\beta, w = g^{\alpha_0}, Z = e(g,g)^{\alpha a_1 b}, g^{\boldsymbol{\alpha}}, g^{a_1}, g^{a_2}, g^b, g^{ba_1}, g^{ba_2}, \tau_1 = v \cdot v_1^{a_1}, \tau_2 = v \cdot v_2^{a_2}, T_1 = \tau_1^b, T_2 = \tau_2^b$$

Then it computes the master secret key as:

$$D_1 = g^{\alpha a_1} \cdot v^r, \quad D_2 = g^{-\alpha + z_1} \cdot v_1^r, \quad D_3 = g^{-bz_1}, \quad D_4 = v_2^r \cdot g^{z_2}, D_5 = g^{-bz_2}, \quad D_6 = g^{br_2},$$
$$D_7 = g^{r_1}, \quad K_0 = g^{r_1(<\boldsymbol{x_0}, \boldsymbol{\alpha}> + \alpha_0 tag_V + \beta)}, \quad \boldsymbol{K} = g^{r_1(\boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_V)}$$

where $r = r_1 + r_2$. The master secret key is defined to be $K_\top = K_V = (D_1, \ldots, D_7, K_0, \boldsymbol{K}, tag_V, \boldsymbol{tag_V})$, which can also be considered as the secret key of the whole affine space $V = S(I, \boldsymbol{x_0})$, where I is the identity matrix.

Delegate(PP, V_1, K_{V_1}, V_2): The algorithm takes input two affine subspaces $V_1 = S(M_1, \boldsymbol{x_1})$, $V_2 = S(M_2, \boldsymbol{x_2})$ and a secret key of the affine subspace V_1 with the form:

$$K_{V_1} = \begin{pmatrix} D_1 = g^{\alpha a_1} \cdot v^r, \quad D_2 = g^{-\alpha + z_1} \cdot v_1^r, \quad D_3 = g^{-bz_1}, \quad D_4 = v_2^r \cdot g^{z_2}, \\ D_5 = g^{-bz_2}, \quad D_6 = g^{br_2}, \quad D_7 = g^{r_1}, \\ K_0 = g^{r_1(<\boldsymbol{x_1}, \boldsymbol{\alpha}> + \alpha_0 tag_{V_1} + \beta)}, \quad \boldsymbol{K} = g^{r_1(M_1^\top \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_1})}, \quad tag_{V_1}, \quad \boldsymbol{tag}_{V_1} \end{pmatrix}$$

Since $V_2 \subseteq V_1$, we must have $M_2 = M_1 \cdot T$ and $\boldsymbol{x_2} = \boldsymbol{x_1} + M_1^\top \cdot \boldsymbol{y}$ for some efficiently computable matrix T and vector $\boldsymbol{y}$. We can then compute a key $K'_{V_2} = (D'_1, \ldots, D'_7, K'_0, \boldsymbol{K'}, tag_{V_2}, \boldsymbol{tag}_{V_2})$ for V_2:

$$D'_1 = D_1, D'_2 = D_2, D'_3 = D_3, D'_4 = D_4, D'_5 = D_5, D'_6 = D_6, D'_7 = D_7,$$
$$K'_0 = g^{r_1(<\boldsymbol{x_1}, \boldsymbol{\alpha}> + \alpha_0 tag_{V_1} + \beta)} \cdot g^{r_1 \boldsymbol{y}^\top \cdot (M_1^\top \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_1})} = g^{r_1(<\boldsymbol{x_2}, \boldsymbol{\alpha}> + \alpha_0 tag_{V_2} + \beta)},$$
$$\boldsymbol{K'} = g^{r_1 T^\top \cdot (M_1^\top \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_1})} = g^{r_1(M_2^\top \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_2})}$$

and the tag of K'_{V_2} are $tag_{V_2} = tag_{V_1} + < M_1^\top \cdot \boldsymbol{y}, \boldsymbol{tag}_{V_1} >$, $\boldsymbol{tag}_{V_2} = T^\top \cdot \boldsymbol{tag}_{V_1}$.

However, we also need to re-randomize the new secret key to ensure all secret keys for the same affine subspace having the same distribution. To do this, it randomly picks $r'_1, r'_2, z'_1, z'_2 \in \mathbb{Z}_p$ and computes

$$D''_1 = D'_1 \cdot v^{r'_1 + r'_2} = g^{\alpha a_1} \cdot v^{r''}, \quad D_2 = D'_2 \cdot g^{z'_1} \cdot v_1^{r'_1 + r'_2} = g^{-\alpha + z''_1} \cdot v_1^{r''},$$
$$D''_3 = D'_3 \cdot g^{-bz'_1} = g^{-bz''_1}, \quad D''_4 = D'_4 \cdot v_2^{r'_1 + r'_2} \cdot g^{z'_2} = v_2^{r''} \cdot g^{z''_2},$$
$$D''_5 = D'_5 \cdot g^{-bz'_2} = g^{-bz''_2}, \quad D''_6 = D'_6 \cdot g^{br'_2} = g^{br''_2}, \quad D''_7 = D'_7 \cdot g^{r'_1} = g^{r''_1},$$
$$K''_0 = g^{r_1(<\boldsymbol{x_2}, \boldsymbol{\alpha}> + \alpha_0 tag_{V_2} + \beta)} \cdot g^{r'_1(<\boldsymbol{x_2}, \boldsymbol{\alpha}> + \alpha_0 tag_{V_2} + \beta)} = g^{r''_1(<\boldsymbol{x_1}, \boldsymbol{\alpha}> + \alpha_0 tag_{V_2} + \beta)},$$
$$\boldsymbol{K''} = g^{r_1(M_2^\top \cdot \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_2})} \cdot g^{r'_1(M_2^\top \cdot \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_2})} = g^{r''_1(M_2^\top \boldsymbol{\alpha} + \alpha_0 \boldsymbol{tag}_{V_2})}$$

where $z_1'' = z_1 + z_1', z_2'' = z_2 + z_2', r_1'' = r_1 + r_1', r_2'' = r_2 + r_2', r'' = r + r_1' + r_2'$. And it outputs the secret key $K_{V_2} = (D_1'', \ldots, D_7'', K_0'', \boldsymbol{K''}, tag_{V_2}, \boldsymbol{tag}_{V_2})$.

Encrypt$(PP, \mathbf{x}, m)$: Given a message $m \in \mathbb{G}_T$ and a vector $\boldsymbol{x} \in V$, the encryption algorithm randomly chooses $s_1, s_2, t, tag_c \in \mathbb{Z}_p$ and computes

$$C_0 = m \cdot Z^{s_2}, \quad C_1 = g^{b(s_1+s_2)}, \quad C_2 = g^{ba_1 s_1}, \quad C_3 = g^{a_1 s_1}, \quad C_4 = g^{ba_2 s_2}, \quad C_5 = g^{a_2 s_2},$$
$$C_6 = \tau_1^{s_1} \cdot \tau_2^{s_2}, \quad C_7 = T_1^{s_1} \cdot T_2^{s_2} \cdot w^{-t}, \quad E_1 = (g^{\alpha_0 \cdot tag_c + <\boldsymbol{x}, \boldsymbol{\alpha}> + \beta})^t, \quad E_2 = g^t,$$

And outputs $CT_{\boldsymbol{x}} = (C_0, C_1, \ldots, C_7, E_1, E_2, tag_c)$.

Decrypt$(PP, CT_{\boldsymbol{x}}, K_{V'}, \boldsymbol{x})$: If $\boldsymbol{x} \in V' = S(M, \boldsymbol{x'})$, we can efficiently find $\boldsymbol{y}$ such that $\boldsymbol{x} = \boldsymbol{x'} + M^\top \cdot \boldsymbol{y}$. If $< \boldsymbol{y}, \boldsymbol{tag}_{V'} > + tag_{V'} - tag_c \neq 0$, it recovers

$$\phi_1 = (\prod_{j=1}^{5} e(C_j, D_j)) \cdot (\prod_{j=6}^{7} e(C_j, D_j))^{-1} = e(g,g)^{\alpha a_1 b s_2} \cdot e(g,w)^{r_1 t}$$

$$\phi_2 = \left(\frac{e(\boldsymbol{K}^{\mathbf{y}} K_0, E_2)}{e(E_1, D_7)} \right)^{\frac{1}{<\boldsymbol{y}, \boldsymbol{tag}_{V'}> + tag_{V'} - tag_c}} = e(g,w)^{r_1 t}$$

It finally recovers the plaintext as

$$m = E_0 \cdot \phi_2 \cdot \phi_1^{-1}$$

Otherwise, the algorithm aborts and returns $\perp$.

The Independence of Delegation. In the definition, we also require that delegation is independent of the path taken. That is, for the affine subspaces $V_3 = S(M_3, \boldsymbol{x}_3), V_2 = S(M_2, \boldsymbol{x}_2), V_1 = S(M_1, \boldsymbol{x}_1)$ satisfying $V_3 \subseteq V_2 \subseteq V_1$, $Delegate(PP, V_1, K_{V_1}, V_3)$ should produce the same distribution as $Delegate(PP, V_2, Delegate(PP, V_1, K_{V_1}, V_2), V_3)$. From the denotation of affine space, we have:

$$M_3 \cdot T_3 = M_1 = M_2 \cdot T_2, \quad M_3 \cdot T_3' = M_2$$
$$\boldsymbol{x}_3 = M_2^\top \cdot \boldsymbol{y} + \boldsymbol{x}_2, \quad \boldsymbol{x}_2 = M_1^\top \cdot \boldsymbol{y'} + \boldsymbol{x}_2, \quad \boldsymbol{x}_3 = M_1^\top \cdot \boldsymbol{y''} + \boldsymbol{x}_1.$$

In our scheme, suppose two secret keys of subspace V_3 are generated from different delegations as following:

$$K_{V_3} = (D_1, \ldots, D_7, K_0, \boldsymbol{K}, tag_{V_3}, \boldsymbol{tag}_{V_3}) \leftarrow Delegate(PP, V_1, K_{V_1}, V_3)$$
$$K_{V_3}' = (D_1', \ldots, D_7', K_0', \boldsymbol{K}', tag_{V_3}', \boldsymbol{tag}_{V_3}') \leftarrow Delegate(PP, V_2, Delegate(PP, V_1, K_{V_1}, V_2), V_3)$$

where $D_1, \ldots, D_7$ and $D_1', \ldots, D_7'$ have the same distribution because of the re-randomizing. Due to the way of tag generation in the delegation, we have the relationship:

$$\begin{aligned} \boldsymbol{tag}_{V_3}' &= (T_3')^\top \cdot \boldsymbol{tag}_{V_2} = (T_3')^\top \cdot T_2^\top \cdot \boldsymbol{tag}_{V_1} = T_3^\top \cdot \boldsymbol{tag}_{V_1} = \boldsymbol{tag}_{V_3}, \\ tag_{V_3} &= tag_{V_2} + < M_2^\top \cdot \boldsymbol{y}, \boldsymbol{tag}_{V_2} > \\ &= tag_{V_1} + < M_1^\top \cdot \boldsymbol{y'}, \boldsymbol{tag}_{V_1} > + < M_2^\top \cdot \boldsymbol{y}, T_2^\top \cdot \boldsymbol{tag}_{V_1} > \\ &= tag_{V_1} + < M_1^\top \cdot \boldsymbol{y''}, \boldsymbol{tag}_{V_1} > = tag_{V_3}' \end{aligned}$$

So the tags from different delegation are equal. With this conclusion, $K_0, \boldsymbol{K}$ and $K'_0, \boldsymbol{K'}$ are also have the same distribution. The two secret keys will have the same distribution and the delegation algorithm coincides with the definition.
Remarks. Now we analysis the probability of the abortion in the decryption algorithm. Since the tags of the secret key are independent from different delegations, we have

$$\begin{aligned} &< \boldsymbol{y}, \boldsymbol{tag}_{V'} > + tag_{V'} - tag_c \\ &= < \boldsymbol{y}, (T^{\top})^{-1} \cdot \boldsymbol{tag}_V > + tag_V + < \boldsymbol{y'}, \boldsymbol{tag}_V > - tag_c \\ &= < \boldsymbol{y''}, \boldsymbol{tag}_V > + tag_V - tag_c \end{aligned}$$

where $\boldsymbol{y''} = \boldsymbol{y} + (T^{\top})^{-1} \cdot \boldsymbol{y'}$ and $\boldsymbol{x'} = I^{\top} \cdot \boldsymbol{y'}$. Since $\boldsymbol{tag}_V, tag_V$ are uniform distributed in their domains, the algorithm aborts with $1/p$ probability.

Theorem 1. *The construction above is adaptively secure under the DLIN and DBDH assumptions.*

Proof. The proof uses the dual system methodology introduced in [16], which involves ciphertexts and private keys that can be normal or semi-functional.

- Semi-functional ciphertexts are generated by first computing a normal ciphertext $CT_{\mathbf{x}} = (C_0, C_1, \ldots, C_7, E_1, E_2, tag_c)$. Then it chooses a random $x \in \mathbb{Z}_p$. It sets $C'_0 = C_0, C'_1 = C_1, C'_2 = C_2, C'_3 = C_3, E'_1 = E_1, E'_2 = E_2, tag'_c = tag_c$, leaving these elements and the tag unchanged. It then sets

  $$C'_4 = C_4 \cdot g^{ba_2x}, C'_5 = C_5 \cdot g^{a_2x}, C'_6 = C_6 \cdot v_2^{a_2x}, C'_7 = C_7 \cdot v_2^{ba_2x}$$

  The semi-functional ciphertext is $CT'_{\boldsymbol{x}} = (C'_0, C'_1, \ldots, C'_7, E'_1, E'_2, tag'_c)$.
- Semi-functional secret keys are generated by first computing a normal secret key $K_V = (D_1, \ldots, D_7, K_0, \boldsymbol{K}, tag_V, \boldsymbol{tag}_V)$. Then it chooses a random $\gamma \in \mathbb{Z}_p$. It sets $D'_3 = D_3, D'_5 = D_5, D'_6 = D_6, D'_7 = D_7, K'_0 = K_0, \boldsymbol{K'} = \boldsymbol{K}, tag'_V = tag_V, \boldsymbol{tag'}_V = \boldsymbol{tag}_V$, leaving these elements and the tag unchanged. It then sets

  $$D'_1 = D_1 \cdot g^{-a_1a_2\gamma}, D'_2 = D_2 \cdot g^{a_2\gamma}, D'_4 = D_4 \cdot g^{a_1\gamma}$$

  The semi-functional secret key is $K'_V = (D'_1, \ldots, D'_7, K'_0, \boldsymbol{K'}, tag'_V, \boldsymbol{tag'}_V)$.

The proof proceeds with a game sequence starting from $Game_{Real}$, which is the actual attack game. The following games are defined below.

$Game_0$ is the real attack game but the challenge ciphertext is semi-functional.

$Game_k$ (for $1 \leq k \leq q$) is identical to $Game_0$ except that the first k secret key delegation queries are answered by returning semi-functional secret keys.

$Game_{q+1}$ is as $Game_q$ but the challenge ciphertext is a semi-functional encryption of a random element of $\mathbb{G}_T$ instead of the actual plaintext.

We prove the indistinguishability between two consecutive games under some assumptions below. The sequence ends in q+1, where the challenge ciphertext is independent of the challenger's bit μ, hence any adversary has no advantage. □

Lemma 1. *If DLIN is hard, $Game_0$ is indistinguishable from $Game_{Real}$.*

Proof. The simulator $\mathcal{S}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision linear problem. We now describe how it executes the setup, delegate phases, and challenge phase of the spatial encryption game with the adversary $\mathcal{A}$.

Setup. The algorithm chooses random exponents $b, \alpha, y_v, y_{v_1}, y_{v_2} \in \mathbb{Z}_p$ and random group elements $g^{\alpha_1}, \ldots, g^{\alpha_n}, g^{\beta}, w \in \mathbb{G}$. It then implicitly sets $g = g, g^{a_1} = f, g^{a_2} = \nu$.

Finally, it sets the variables as: $g^b, g^{ba_1} = f^b, g^{ba_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}$.

Using this it can calculate τ_1, τ_2, T_1, T_2 and $e(g,g)^{\alpha a_1 b} = e(g,f)^{\alpha \cdot b}$ in order to publish the public parameters PP. We also note that using α it can compute the master secret key.

Key Delegation Phases 1,2. Since simulator $\mathcal{S}$ has the actual master secret key K_V it simply runs the delegation algorithm to generate the keys in both phases. Note that the K_V it has only allows for the creation of normal keys.

Challenge. The simulator $\mathcal{S}$ two messages m_0, m_1 and challenge vector $\mathbf{x}$. It then flips a coin μ. We describe the creation of the challenge ciphertext in two steps. First, it creates a normal ciphertext using the real algorithm by calling **Encrypt**$(PP, \mathbf{x}, m_\mu)$, which outputs a ciphertext $CT'_{\mathbf{x}} = (C'_0, C'_1, \ldots, C'_7, E'_1, E'_2, tag'_c)$. Let s'_1, s'_2, t' be the random exponents used in creating the ciphertext. Then we modify the components of the ciphertext as follows. It sets

$$C_0 = C'_0 \cdot (e(g^{c_1}, f) \cdot e(g, f^{c_2}))^{b\alpha},\ C_1 = C'_1 \cdot g^{bc_1},\ C_2 = C'_2 \cdot f^{-bc_2},\ C_3 = C'_3 \cdot f^{c_2},\ C_4 = C'_4 \cdot T^b,\ C_5 = C'_5 \cdot T,$$

$$C_6 = C'_6 \cdot g^{y_v c_1} \cdot f^{-y_{v_1} c_2} \cdot T^{y_{v_2}},\ C_7 = C'_7 \cdot (g^{y_v c_1} \cdot f^{-y_{v_1} c_2} \cdot T^{y_{v_2}})^b,\ E_1 = E'_1,\ E_2 = E'_2,\ tag_c = tag'_c$$

where this assignment implicitly sets $s_1 = -c_2 + s_1, s_2 = s'_2 + c_1 + c_2$, and $s = s'_1 + s'_2 + c_1$. The returned ciphertext is $CT^*_{\mathbf{x}} = (C_0, C_1, \ldots, C_7, E_1, E_2, tag_c)$. If $T = \nu^{c_1 + c_2}$, it will have the same distribution as a standard ciphertext; otherwise, it will be distributed identically to a semi-functional ciphertext. The simulator S receives a bit μ' and outputs 0, if $\mu' = \mu$. □

Lemma 2. *For any $1 \leq k \leq q$, if an adversary $\mathcal{A}$ can distinguish $Game_k$ from $Game_{k-1}$, we can build a distinguisher for the DLIN problem.*

Proof. In this proof, the simulator $\mathcal{S}$ can create semi-functional keys for any affine spaces. However, the simulator $\mathcal{S}$ can not simply create an affine space V_k containing the challenged vector deciding whether the k-th queried private key is normal or semi-functional. Since in the reduction we create the tags for the k-th secret key and the challenged ciphertext with some linear relations. With this relation, the simulator $\mathcal{S}$ can not create a secret key that can decrypt the challenged ciphertext independently of whether it was a semi-functional secret key. But the linear relations information theoretically hidden to the adversary. That is, in the view of the adversary, the tag of the k-th secret key and the tag of ciphertext are completely independent.

The simulator $\mathcal{S}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision linear problem. We now describe how it executes the setup, delegate phases, and challenge phase of the spatial encryption game with the adversary $\mathcal{A}$.

Setup. The simulator $\mathcal{S}$ picks $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_u, y_h$. It then sets

$$g = g, Z = e(g,f)^{\alpha a_1}, g^{a_1}, g^{a_2}, g^b = f, g^{ba_1} = (f)^{a_1}, g^{ba_2} = (f)^{a_2}, v = \nu^{-a_1 \cdot a_2}$$

$$v_1 = \nu^{a_2} \cdot g^{y_{v_1}}, v_2 = \nu^{a_1} \cdot g^{y_{v_2}}, \tau_1 = g^{y_{v_1} a_1}, \tau_2 = g^{y_{v_2} a_2}, T_1 = f^{y_{v_1} a_1}, T_2 = f^{y_{v_2} a_2}$$

Finally, $\mathcal{S}$ randomly chooses $A_0, B_0, \alpha'_0 \in \mathbb{Z}_p$, $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{Z}_p^n$ and sets

$$g^{\boldsymbol{\alpha}} = f^{\boldsymbol{A}} \cdot g^{\boldsymbol{B}}, g^{\beta} = f^{A_0} \cdot g^{B_0}, g^{\alpha_0} = f \cdot g^{\alpha'_0}$$

This will define all the public parameters of the system. Note that by virtue of knowing α, the simulator $\mathcal{S}$ will know the regular master secret key.

Key Delegation Phases 1,2. We break the key delegation queries into three cases. Secret key delegation is done the same regardless of whether we are in phase 1 or 2. Consider the i-th query made by adversary $\mathcal{A}$.

- **Case 1:** $i > k$ When i is greater than k, the simulator $\mathcal{S}$ will generate a normal key for the delegation query of affine subspace V_i. Since it has the master secret key K_V it can run that algorithm.
- **Case 2:** $i < k$ When i is less than k, the simulator $\mathcal{S}$ will generate a semi-functional key for the delegation query of affine subspace V_i. It first creates a normal key using K_V. Then it makes it semi-functional using $g^{a_1 a_2}$.
- **Case 3:** $i = k$ The algorithm first runs the secret key delegation algorithm to generate a normal secret key K_{V_k} with $D'_1, \ldots, D'_7, K'_0, \boldsymbol{K'}$ using

  $$tag_{V_k} = - < \boldsymbol{A}, \boldsymbol{x}_k > - A_0, \qquad \boldsymbol{tag}_{V_k} = -M_k^{\top} \cdot \boldsymbol{A}$$

  for the requested affine subspace $V_k = S(M_k, \mathbf{x}_k)$. It implicitly sets the tag of the master secret key $\boldsymbol{tag}_V = \boldsymbol{A}$. Let r'_1, r'_2, z'_1, z'_2 be random exponents used.

  $$D_1 = D'_1 \cdot T^{-a_1 a_2},\ D_2 = D'_2 \cdot T^{a_2} (g^{c_1})^{y_{v_1}},\ D_3 = D'_3 \cdot (f^{c_2})^{y_{v_2}},\ D_4 = D'_4 \cdot T^{a_1} (g^{c_1})^{y_{v_2}},\ D_5 = D'_5 \cdot (f^{c_2})^{y_{v_2}},$$
  $$D_6 = D'_6 \cdot f^{c_2},\ D_7 = D'_7 \cdot g^{c_1},\ K_0 = K'_0 \cdot (g^{c_1})^{<\boldsymbol{B}, \boldsymbol{x}_k> + B_0 - (<\boldsymbol{A}, \boldsymbol{x}_k> + A_0)\alpha'_0},\ \boldsymbol{K} = \boldsymbol{K'} \cdot (g^{c_1})^{M_k^{\top} \cdot \boldsymbol{B} - \alpha'_0 M_k^{\top} \cdot \boldsymbol{A}}$$

  The semi-functional secret key is $K_{V_k} = (D_1, \ldots, D_7, K_0, \boldsymbol{K}, tag_{V_k}, \boldsymbol{tag}_{V_k})$. In addition, we note that we implicitly set $z_1 = z'_1 - y_{v_1} c_2$, $z_2 = z'_2 - y_{v_2} c_2$, $r_1 = r'_1 + c_1$ and $r_2 = r'_2 + c_2$. If T is a linear tuple of the form $T = \nu^{c_1 + c_2}$, then the k-th query results in a normal key. Otherwise, if T is a random group element, then we can write $T = \nu^{c_1 + c_2} g^{\gamma}$ for random $\gamma \in \mathbb{Z}_p$. This forms a semi-functional key where γ is the added randomness to make it semi-functional.

Challenge. The simulator $\mathcal{S}$ is given a challenge vector $\boldsymbol{x}$ and the messages m_0, m_1. Then it flips a coin μ. In this phase $\mathcal{S}$ needs to be able to generate a semi-functional challenge ciphertext. One problem is that $\mathcal{S}$ does not have the group element v_2^b so it cannot directly create such a ciphertext. However, in the case where $tag_c = - < \boldsymbol{x}, \boldsymbol{A} > - A_0$ it will have a different method of doing so.

$\mathcal{S}$ first runs the normal encryption algorithm to generate a normal ciphertext CT' for vector $\boldsymbol{x}$ and message m_μ with a tag $tag_c = - < \boldsymbol{x}, \boldsymbol{A} > - A_0$. It then gets a standard ciphertext $C'_0, C'_1, \ldots, C'_7, E'_1, E'_2$ under random exponents s'_1, s'_2, t' and sets

$$C_0 = C'_0,\ C_1 = C'_1,\ C_2 = C'_2,\ C_3 = C'_3,\ C_4 = C'_4 \cdot f^{a_2 \delta},\ C_5 = C'_5 \cdot g^{a_2 \delta},\ C_6 = C'_6 \cdot v^{a_2 \delta},$$

$$C_7 = C'_7 \cdot f^{y_{v_2} \delta a_2} \nu^{-a_1 \delta \alpha'_0 a_2},\ E_1 = E'_1 \cdot (\nu^{<\boldsymbol{B}, \boldsymbol{x}> + B_0 - (<\boldsymbol{x}, \boldsymbol{A}> + A_0)\alpha'_0})^{a_1 a_2 \delta},\ E_2 = E'_2 \cdot \nu^{a_1 a_2 \delta}$$

If T is a tuple, then we are in $Game_{k-1}$, otherwise we are in $Game_k$.

Note that, the simulator can not generate the secret keys to decrypt challenged ciphertext neither it is normal or semi-functional. Because for a vector subspace $V' = S(M', \boldsymbol{x'})$ containing $\boldsymbol{x}$, the tag of $K_{V'}$ generated by $\mathcal{S}$ is $tag_{V'} = -<\boldsymbol{A}, \boldsymbol{x'}> -A_0$, $\boldsymbol{tag}_{V'} = -(M')^\top \cdot \boldsymbol{A}$. And for some $\boldsymbol{y}$, we have $\boldsymbol{x} = (M')^\top \cdot \boldsymbol{y} + \boldsymbol{x'}$ and $<\boldsymbol{tag}_{V'}, \boldsymbol{y}> + tag_{V'} - tag_c = 0$ Due to the decryption algorithm, the secret key can not decrypt the challenged ciphertext even it is normal form.

Since $(tag_{V_k}, \boldsymbol{tag}_{V_k})$ and tag_c are independent in the scheme, we should clarify that the adversary can not detect any special relationship between $tag_{V_k}, \boldsymbol{tag}_{V_k}, tag_c$ in the simulation. Suppose they are linearly dependent, that is, there exists constants $\zeta, \eta \in \mathbb{Z}_p$, $\boldsymbol{\zeta} \in \mathbb{Z}_p^n$ and we have the relation $\zeta \cdot tag_{V_k} + \eta \cdot tag_c + <\boldsymbol{\zeta}, \boldsymbol{tag}_{V_k}> = 0$. We simplifies it by

$$(\zeta - \eta)A_0 + < M_k^\top \cdot \boldsymbol{\zeta} + \zeta \boldsymbol{x}_k - \eta \boldsymbol{x}, \boldsymbol{A} > = 0$$

This implies, for some $\boldsymbol{y} = \boldsymbol{\zeta}/\eta$, we have $\boldsymbol{x} = \boldsymbol{x_k} + M_k^\top \cdot \boldsymbol{y}$. It conflicts with the definition of the game, because the vector $\boldsymbol{x}$ is in the subspace V_k. So $tag_{V_k}, \boldsymbol{tag}_{V_k}, tag_c$ are linearly independent, and $A_0, \boldsymbol{A}$ are hidden from the adversary's view.

$\mathcal{S}$ receives a bit μ' and outputs 0 if $\mu' = \mu$. $\square$

Lemma 3. *Suppose that there exists an adversary $\mathcal{A}$ that makes at most q queries and $|Game_q - Game_{q+1}| = \epsilon$. Then we can build a simulator $\mathcal{S}$ that has advantage ϵ in solving the DBDH problem.*

Proof. We begin by noting that in both of these two games the challenge ciphertexts and all the secret keys are semi-functional. Therefore, the simulator $\mathcal{S}$ only needs to be able to generate semi-functional secret keys. $\mathcal{S}$ begins by taking in a $DBDH$ instance $(\mathbb{G}, g, g^{c_1}, g^{c_2}, g^{c_3}, T)$.

Setup. The simulator $\mathcal{S}$ begins by choosing random exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, \alpha_0, \alpha_1, \ldots, \alpha_n, \beta \in \mathbb{Z}_p$. It then sets

$$g = g, g^\beta, Z = e(g^{c_1}, g^{c_2}), g^{\boldsymbol{\alpha}} = (g^{\alpha_1}, \ldots, g^{\alpha_n}), g^{a_1}, g^{a_2} = g^{c_2}, g^b, g^{ba_1}, g^{ba_2} = g^{bc_2},$$
$$v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}, w = g^{\alpha_0}, \tau_1 = v \cdot v_1^{a_1}, \tau_1^b, \tau_2 = v \cdot (g^{c_1})^{y_{v_2}}, \tau_2^b$$

The simulator $\mathcal{S}$ publish the public key PK. Note that the master secret key g^α is not available to S. We point out that this setup lets $\alpha = c_1 \cdot c_2, a_2 = c_2$.

Key Delegation Phase 1,2. All secret key delegations result in semi-functional keys. When a request for an affine subspace $V_1 = S(M, \boldsymbol{x_1})$ is made, the secret key delegation algorithm chooses random $r_1, r_2, z_1, z_2, \gamma', tag_{V_1}, \boldsymbol{tag}_{V_1}$ and defines $r = r_1 + r_2$. It implicitly sets the variable $\gamma = \gamma' + c_1$.

It creates the key as:

$$D_1 = (g^{c_2})^{-\gamma' a_1} v^r,\ D_2 = (g^{c_2})^{-\gamma'} v_1^r g^{z_1},\ D_3 = (g^b)^{-z_1},\ D_4 = (g^{c_1})^{a_1} g^{a_1 \gamma'} v_2^r g^{z_2},$$
$$D_5 = (g^b)^{-z_2},\ D_6 = g^{r_2 b}, D_6 = g^{r_1},\ K_0 = g^{r_1(<\boldsymbol{x_1}, \boldsymbol{a}> + \alpha_0 tag_{V_1} + \beta)},\ \boldsymbol{K} = g^{r_1(M^\top \cdot \boldsymbol{a} + \alpha_0 \boldsymbol{tag}_{V_1})}$$

Challenge. The simulator $\mathcal{S}$ receives a challenge vector $\boldsymbol{x}$ and two message m_0, m_1 from the attacker. $\mathcal{S}$ will now create a challenge ciphertext that is a semi-functional ciphertext of either m_μ or a random message, depending on T. It first chooses a random bit μ.

$\mathcal{S}$ chooses random s_1, t and $tag_c \in \mathbb{Z}_p$. It will implicitly let $s_2 = c_3$. The message $m_\mu \in \mathbb{G}_T$ is blinded as $C_0 = m_\mu \cdot T^{a_1 b}$. It then chooses random $x' \in \mathbb{Z}_p$ and will implicitly set $x = x' - c_3$.

$$C_1 = g^{s_1 b} \cdot (g^{c_3})^b,\ C_2 = g^{ba_1 s_1},\ C_3 = g^{bs_1},\ C_4 = (g^{c_2})^{x' b},\ C_5 = (g^{c_2})^{x'},$$

$$C_6 = \tau_1^{s_1}(g^{c_3})^{y_v}(g^{c_2})^{y_{v_2} x'},\ C_7 = \tau_1^{bs_1}(g^{c_3})^{by_v}(g^{c_2})^{y_{v_2} x' b} w^{-t},\ E_1 = (g^{\alpha_0 \cdot tag_c + <\boldsymbol{x},\boldsymbol{a}> + \beta})^t,\ E_2 = g^t$$

If T is a tuple, then we are in $Game_q$, otherwise we are in $Game_{q+1}$. $\mathcal{S}$ receives a bit μ' and outputs 0 if $\mu = \mu'$. □

5 Adaptively Secure Doubly-Spatial Encryption

In this section, we propose an adaptively secure doubly-spatial encryption. In the doubly-spatial encryption, the policies, as well as roles, are the affine subspaces in $\mathbb{Z}_p^n$. The roles and policies are opposites of each other in same way. That is, $\mathbb{Z}_p^n$ is the most powerful role, but the weakest policy, and vectors are the strictest policies, but the most restricted roles.

Our construction is not so efficient as the spatial encryption construction, because the length of the ciphertext depends on the dimension of the affine space. We change the tag attaching in the ciphertext by a vector $(tag_c, \boldsymbol{tag}_c) \in \mathbb{Z}_p^{n+1}$. Then the ciphertext and the secret key can be seen as a dual pair in the construction. The scheme consists four algorithms, and the **Setup** and **Delegate** algorithms are completely the same as the spatial encryption scheme in **Sec.** 4. So we omit them in this paper.

Encrypt(PP, W, m): Given a message $m \in \mathbb{G}_T$ and an affine subspace $W = S(M, \boldsymbol{x})$, the encryption algorithm randomly chooses $s_1, s_2, t, tag_c \in \mathbb{Z}_p$, $\boldsymbol{tag}_c \in \mathbb{Z}_p^n$ and computes

$$C_0 = m \cdot Z^{s_2},\ C_1 = g^{b(s_1+s_2)},\ C_2 = g^{ba_1 s_1},\ C_3 = g^{a_1 s_1},\ C_4 = g^{ba_2 s_2},\quad C_5 = g^{a_2 s_2},\ C_6 = \tau_1^{s_1} \cdot \tau_2^{s_2},$$
$$C_7 = T_1^{s_1} \cdot T_2^{s_2} \cdot w^{-t},\ E_1 = (g^{\alpha_0 \cdot tag_c + <\boldsymbol{x},\boldsymbol{a}> + \beta})^t,\ E_2 = g^t,\ \boldsymbol{E}_3 = (g^{\alpha_0 \boldsymbol{tag}_c + M^\top \boldsymbol{a}})^t$$

And outputs $CT_W = (C_0, C_1, \ldots, C_7, E_1, E_2, \boldsymbol{E}_3, tag_c, \boldsymbol{tag}_c)$.

Decrypt$(PP, CT_W, K_{V'}, W)$: If $V' \cap W \neq \emptyset$, there exits a vector $\boldsymbol{x}^* \in V' \cap W$. Then we can efficiently find $\boldsymbol{y}, \boldsymbol{y'}$ such that $\boldsymbol{x}^* = \boldsymbol{x} + M^\top \cdot \boldsymbol{y} = \boldsymbol{x'} + (M')^\top \cdot \boldsymbol{y'}$, where $W = S(M, \boldsymbol{x}), V' = S(M', \boldsymbol{x'})$. If $< \boldsymbol{tag}_{V'}, \boldsymbol{y'} > + tag_{V'} - (tag_c + < \boldsymbol{tag}_c, \boldsymbol{y} >) \neq 0$, it then recovers

$$\phi_1 = (\prod_{j=1}^{5} e(C_j, D_j)) \cdot (\prod_{j=6}^{7} e(C_j, D_j))^{-1} = e(g,g)^{\alpha a_1 b s_2} \cdot e(g,w)^{r_1 t}$$

$$\phi_2 = \left(\frac{e(\boldsymbol{K}^{\boldsymbol{y'}} K_0, E_2)}{e(\boldsymbol{E_3}^{\boldsymbol{y}} E_1, D_7)} \right)^{\frac{1}{<\boldsymbol{tag}_{V'}, \boldsymbol{y'}> + tag_{V'} - (tag_c + <\boldsymbol{tag}_c, \boldsymbol{y}>)}} = e(g,w)^{r_1 t}$$

It finally recovers the plaintext as

$$m = E_0 \cdot \phi_2 \cdot \phi_1^{-1}$$

Otherwise, the algorithm aborts and returns $\perp$. And the algorithm aborts with $1/p$ probability.

Theorem 2. *The doubly-spatial encryption construction is adaptively secure under the DLIN and DBDH assumptions.*

Due to space considerations the proof is given briefly in the appendix.

6 Conclusion

We give a fully secure spatial and a fully secure doubly-spatial encryption scheme over prime order groups under standard assumptions, the decisional linear (DLIN) assumption and the decisional bilinear Diffe-Hellman (DBDH) assumption, in the standard model. To the best of our knowledge, no presented correlated work has achieved this. However, how to construct (doubly-)spatial encryption which has strong anonymous property is still an open problem.

References

1. Attrapadung, N., Libert, B.: Functional encryption for inner product achieving constant-size ciphertexts with adaptive security or support for negation. In: P.Q. Nguyen, D. Pointcheval (Eds.) PKC 2010, LNCS 6056, pp. 384-402. Springer, 2010
2. Boneh, D., Hamburg, M.: Generalized identity based and broadcast encryption schemes. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 455-470. Springer, 2008
3. Canetti, R., Goldwasser, S.: An efficient threshold public key cryptosystem secure against adaptive chosen ciphertext attack. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 90-106. Springer, 1999
4. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 254-271. Springer, 2003
5. Delerablee, C., Pointcheval, D.: Dynamic threshold public-key encryption. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 317-334. Springer, 2008
6. Freeman, D. M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: H. Gilbert (Ed.) EUROCRYPT 2010, LNCS 6110, pp. 44-61. Springer, 2010
7. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access conrol of encrypted data. In: ACM conference on Computer and Communications Security (ACM CCS), 2006
8. Hamburg, M.: Spatial encryption. IACR Cryptology ePrint Archive 2011: 389
9. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62-91. Springer, 2010
10. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. IEEE Symposium on Security and Privacy 2010: 273-285
11. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455-479. Springer, 2010
12. Moriyama, D., Doi, H.: A fully secure spatial encryption scheme. IEICE Transactions 94-A(1): 28-35 (2011)
13. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191-208. Springer, 2010
14. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457-473. Springer, 2005
15. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47-53. Springer, 1985
16. Waters, B.: Dual system encryption realizing fully secure IBE and HIBE under simple assumptions. In: S. Halevi (Ed.) CRYPTO 2009, LNCS 5677, pp. 619-636. Springer, 2009.
17. Zhou, M., Cao, Z.: Spatial encryption under simpler assumption. In: J. Pieprzyk, F. Zhang (Eds.) ProvSec 2009, LNCS 5848, pp. 19-31. Springer, 2009.

A Proof of Theorem 2

Since the proof of the adaptive security of the doubly-spatial encryption is very alike of the proof in **Sec.** 4, partial proof, which is immediate and trivial from the proof of theorem 1, is omitted here. We briefly give the proof of the indistinguishability of $Game_k$ and $Game_{k+1}$, which is the most non-trivial part in the proof of theorem 2.

The setting of the parameters in the setup phase is the same as lemma 2. And the simulator uses the same way to answer the secret key query as in lemma 2. In the challenge phase, when the simulator S is given a challenge affine subspace $W = S(M, \boldsymbol{x})$ and the messages m_0, m_1, it first runs the normal encryption algorithm to generate a normal ciphertext CT' for W with tag $tag_c = - < \boldsymbol{x}, \boldsymbol{A} > -A_0$ and $\boldsymbol{tag}_c = -M^\top \cdot \boldsymbol{A}$. It then gets a standard ciphertext $C'_0, C'_1, \ldots, C'_7, E'_1, E'_2$ and sets the semi-functional ciphertext as

$$C_0 = C'_0, C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$$

$$C_4 = C'_4 \cdot f^{a_2\delta}, C_5 = C'_5 \cdot g^{a_2\delta}, C_6 = C'_6 \cdot v^{a_2\delta}, C_7 = C'_7 \cdot f^{y_{v_2}\delta a_2} \nu^{-a_1\delta\alpha'_0 a_2}$$

$$E_1 = E'_1 \cdot (\nu^{<\boldsymbol{B},\boldsymbol{x}>+B_0-(<\boldsymbol{x},\boldsymbol{A}>+A_0)\alpha'_0})^{a_1a_2\delta}, E_2 = E'_2 \cdot \nu^{a_1a_2\delta}$$

$$\boldsymbol{E}_3 = \boldsymbol{E}'_3 \cdot (\nu^{M^\top \cdot \boldsymbol{B} - \alpha'_0 M^\top \cdot \boldsymbol{A}})^{a_1a_2\delta}$$

Since $(tag_{V_k}, \boldsymbol{tag}_{V_k})$ and $(tag_c, \boldsymbol{tag}_c)$ are chosen independent in the scheme, we should prove that the adversary cannot detect any special relationship between $tag_{V_k}, \boldsymbol{tag}_{V_k}, tag_c, \boldsymbol{tag}_c$ in the simulation. Suppose they are linearly dependent, there exists constants $\zeta, \eta, \boldsymbol{\zeta}, \boldsymbol{\eta}$ and an equation

$$\zeta \cdot tag_{V_k} + \eta \cdot tag_c + < \boldsymbol{\zeta}, \boldsymbol{tag}_{V_k} > + < \boldsymbol{\eta}, \boldsymbol{tag}_c >= 0$$

This implies, for some $\boldsymbol{y}_1, \boldsymbol{y}_2$, we have $\boldsymbol{x_k} + M_k^\top \cdot \boldsymbol{y}_1 = \boldsymbol{x} + M^\top \cdot \boldsymbol{y}_2$. It means $V_k \cap W \neq \varnothing$. It conflicts with the definition of the game. So we say $tag_{V_k}, \boldsymbol{tag}_{V_k}, tag_c, \boldsymbol{tag}_c$ are linearly independent, and $A_0, \boldsymbol{A}$ are hidden from the adversary's view.