A Tag Based Encoding: An Efficient Encoding for Predicate Encryption in Prime Order Groups

Jongkil Kim[1], Willy Susilo[1], Fuchun Guo[1], and Man Ho Au[2]

[1] Centre of Computer and Information Security Research, School of Computing and Information Technology,
University of Wollongong, Australia,
{jk057, wsusilo, fuchun}@uow.edu.au
[2] The Hong Kong Polytechnic University, Hong Kong,
csallen@comp.polyu.edu.hk

Abstract. We introduce a *tag based encoding*, a new generic framework for modular design of Predicate Encryption (PE) schemes in prime order groups. Our framework is equipped with a compiler which is adaptively secure in prime order groups under the standard Decisional Linear Assumption (DLIN). Compared with prior encoding frameworks in prime order groups which require multiple group elements to interpret a tuple of an encoding in a real scheme, our framework has a distinctive feature which is that each element of an encoding can be represented with only a group element and an integer. This difference allows us to construct a more efficient encryption scheme. In the current literature, the most efficient compiler was proposed by Chen, Gay and Wee (CGW) in Eurocrypt'15. It features one tuple of an encoding into two group elements under the Symmetric External Diffie-Hellman assumption (SXDH). Compared with their compiler, our encoding construction saves the size of either private keys or ciphertexts up-to 25 percent and reduces decryption time and the size of public key up-to 50 percent in 128 security level. Several new schemes such as inner product encryption with short keys, dual spatial encryption with short keys and hierarchical identity based encryption with short ciphertexts are also introduced as instances of our encoding.

Key words: encodings, prime order groups, inner product encryption, spatial encryption, predicate encryption

1 Introduction

Predicate Encryption (PE) is a public key cryptographic system supporting a fine-grained access control. PE schemes have been proposed to support various types of predicates, but many of them share similar features in their constructions and security proofs. Two independent works [2, 32] have been presented by observing the coupling of PE. They formalized common features of PE schemes in composite order groups by encoding predicate parts of the schemes. Those encoding frameworks provide a new direction of proving security since one can show security of a PE scheme by only proving

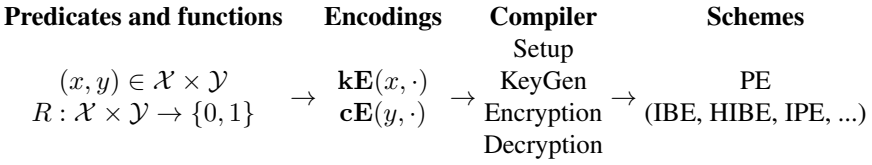

Fig 1: Encoding Frameworks for PE

that an encoding satisfies the syntax required in the framework. Therefore, the encoding frameworks provide a new insight of properties leading to adaptive security.

Despite the advantage, the usage of encoding frameworks [2, 32] were limited since they were introduced only in composite order groups. It is well known that composite order groups significantly harm the efficiency of encryption systems [23, 15, 16]. According to Guillevic [16], to achieve 128 bits security level , the minimum group orders for prime order and composite order bilinear group are 256 and 2,644 bits, resp. Moreover, a pairing computation in composite order groups is about 254 times slower than that of prime order bilinear groups. Hence, constructing adaptively secure PE schemes in prime order groups is desirable to ensure that they are adoptable in practice.

Recently, Chen, Gay and Wee (CGW) presented a dual system attribute based encryption [9] which can be considered as a new compiler in prime order groups for the predicate encoding [32]. They introduced compilers in prime order groups by adopting Dual System Groups (DSG) [10]. In the most efficient compiler of theirs, one composite order group element [32] is represented by two prime order group elements. Independently, Attrapadung [3] and Agrawal and Chase [1] also proposed other compilers in prime order groups, but they showed similar results from an efficiency perspective[3]. All existing compilers show a similar behavior from an efficiency perspective. Specifically, the number of parameters and computation of the resulting scheme in the prime order group is always bounded below by a multiplicative factor, say n, of their counterparts in the composite order groups. The best compiler achieves a factor of $n = 2$ under SXDH assumption in [9, 1] . Moreover, in [3, 9, 1] $n = 3$ is achieved under the DLIN assumption which is weaker than the SXDH assumption. This appears to be the lower bound of the techniques of dual system groups with orthogonal vectors since the size of vectors must be at least 2 to "simulate" the properties of a composite order group. Therefore, it remains an interesting research problem to achieve PE schemes in prime order groups without using vector properties since it may imply more efficient schemes.

1.1 Our Contribution

We introduce a *tag based encoding*, a new generic framework for PE schemes in prime order groups. Compared with prior encoding frameworks in prime order groups, our framework improves the efficiency of prior encodings when the size of an encoding

[3] Attrapadung's compiler [3] needs three group elements for a tuple of an encoding under the DLIN assumption. Agrawal and Chase's compiler [1] requires two group elements under the SXDH assumption and three group elements under the DLIN assumption.

Table 1: An efficiency comparison between our and CGW's compilers [9].

	Assump.	PK	SK	CT
CGW [9]	SXDH	$(2\ell$+3$)\|G_1\|$ + $\|G_T\|$	$2(m_k$+1$)\|G_2\|$	$2(m_c$+1$)\|G_1\| + \|G_T\|$
	DLIN	$(6\ell$+8$)\|G_1\|$ + $2\|G_T\|$	$3(m_k$+1$)\|G_2\|$	$3(m_c$+1$)\|G_1\| + \|G_T\|$
Ours	DLIN	$(\ell + 11)\|G_1\|$ + $\|G_T\|$	$(m_k$+7$)\|G_2\|$ + $m_k\|\mathbb{Z}_p\|$	$(m_c$+8$)\|G_1\| + m_c\|\mathbb{Z}_p\| + \|G_T\|$

	Assump.	PK (by bits)	SK (by bits)	CT (by bits)	Decryption
CGW [9]	SXDH	3840 + 512 ℓ	1024 + 1024 m_k	3584 + 512 m_c	4P + 2 ℓ E
	DLIN	8192 + 1536 ℓ	1536 + 1536 m_k	3840 + 768 m_c	6P + 3 ℓ E
Ours	DLIN	5888 + 256 ℓ	3584+ 768 m_k	5120 + 512m_c	8P+ ℓ E

ℓ: a predicate size (the size of common values in an encoding),
m_k and m_c: the size of encoding schemes used for keys and ciphertexts,
For 128 bits security level [16], we use $|G_1| = |\mathbb{Z}_p| = 256$ bits, $|G_2| = 512$ bits, $|G_T| = 3072$ bits .

scheme is large. Our encoding framework does not use DPVS, DSG or composite order groups. Instead, we utilize tags to construct adaptively secure PE schemes. We observe common properties of PE schemes as other encoding frameworks, but generalize them as a new encoding framework using tag. The generic construction of our encoding is adaptively secure under the Decisional Linear assumption.

Tag based encoding. We introduce a *tag based encoding*. For a predicate R with input domains $\mathcal{X}$ and $\mathcal{Y}$, $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, a tag based encoding for R comprises two algorithms, namely, $\mathbf{kE}$ and $\mathbf{cE}$, together with a field $\mathbb{Z}_p^\ell$ where p is a prime number and ℓ is a value allocated for each function R such as the size of predicate vectors for Inner Product Encryption. We let $\mathbf{kE}(x, \mathbf{h})$ and $\mathbf{cE}(y, \mathbf{h})$ denote the outputs of $\mathbf{kE}$ taking as inputs $x \in \mathcal{X}$ and $\mathbf{h} \in \mathbb{Z}_p^\ell$ and $\mathbf{cE}$ taking as inputs $y \in \mathcal{Y}$ and $\mathbf{h} \in \mathbb{Z}_p^\ell$, respectively. The tag based encoding must satisfy three essential properties, namely Reconstruction, Linearity and $\mathbf{h}$-hiding. Instances of our encoding are interpreted as PE schemes via our constructions. These constructions are often called compilers since they compile encodings to form PE schemes (Fig 1).

An improved efficiency Prior to our work, the most efficient compiler in prime order groups was proposed by Chen, Gay and Wee [9], which is subsequently referred as CGW in this work. The compiler was proposed for the predicate encoding [32]. Multiple compilers under the generalized k-linear assumption [14] were also included in the CGW's framework. The number of group elements that a compiler in the CGW's framework uses to represent a tuple of an encoding (e.g. $\mathbf{kE}$ and $\mathbf{cE}$) depends on computational assumptions of which the compiler is based on. More concretely, each tuple of an encoding scheme is represented by $k + 1$ group elements in private keys and ciphertexts. Also, $k(k + 1)$ elements are required for each coordinate of $\mathbf{h}$ in public keys where $\mathbf{h}$ is a shared input of $\mathbf{kE}$ and $\mathbf{cE}$. The most efficient compiler is under the SXDH assumption (i.e. when k is equal to 1). Two group elements are used for a tuple of an encoding in this compiler. Other encoding frameworks [3, 1] were also proposed independently, but they are similar to the CGW's framework from the efficiency

perspective. Hence, without losing generality, we compare our compiler with CGW's compiler to highlight our contribution.

In our compiler, only one group element is required for each entity of $\mathbf{h}$ in public keys. Hence, if the size of $\mathbf{h}$ is large, our compiler reduces the size of public key to 50 percent compared with the CGW's compiler. Also, it reduces decryption time by 50 percent under the same condition. For the other parameters such as private keys and ciphertexts, our compiler needs a group element and an integer for one tuple of an encoding scheme. The size of the integer in our compiler is the same as the group order of the underlying bilinear group. In other words, it is as small as the size of a group element of G_1 but much less than that of G_2 due to embedding degree of asymmetric bilinear maps. Thus, our compiler reduces the size of either private keys or ciphertexts depending on where G_2 is used for. For example, in 128 bits security level, G_2 requires at least 512 bits. It is twice of the size of $\mathbb{Z}_p$ [16]. It means that only 768 bits are required to represent a tuple in our compiler. This outperforms CGW's approach which require 1024 bits for a tuple. Therefore, our compiler saves the size of private keys or ciphertexts by 25 percent compared to their compiler under the SXDH assumption when the size of an encoding is large.

Moreover, the CGW's framework is also realized under the weaker assumption, namely the DLIN assumption, in comparison to ours[4]. It should be noted that in this setting, 6 group elements are required for public keys for their compiler. It implies that our compiler outperforms their compiler as well in this setting. More concretely, under the same assumption at a 128 bits security level, our compiler saves 83 percent in a public key, 50 percent in private keys, 33 percent in ciphertexts and 66 percent in decryption time if the size of encodings and their shared input is large. We provide table 1 for the details. To compare the efficiency in practice, we compare our inner product encryption with short keys and public attribute inner product encryption to those of other encodings. The instance of Public Attribute Inner Product Encryption (PAIPE) which is taken from [4] is introduced in the appendix. It should be noted that encodings for our IPE schemes are slightly different from those of CGW [9] and Wee [32]. Our instances require one or two fewer elements.

A compiler with symmetric bilinear maps. We also provide a new compiler with symmetric bilinear maps in the appendix. Prior to our works, with symmetric bilinear maps, all encodings [2, 32, 9] are secure only in composite order groups. It is because all prior encodings [9, 3, 1] in prime order groups are based on dual system groups [10] which require asymmetric pairings to feature different properties of left-hand groups and right-hand groups in pairings. To the best of our knowledge, our construction is *the only compiler* that provides adaptive security for encodings with symmetric pairings in prime order groups. This gives our framework an additional flexibility when the encryption scheme is implemented under a special requirement of the pairing type.

New schemes. We introduce a number of new schemes as instances, namely: Inner Product Encryption with short keys, Dual Spatial Encryption with short keys and Hi-

[4] The DLIN assumption with asymmetric bilinear maps can be featured in various forms since it expanded from the DLIN assumption originally equipped with symmetric pairing. The DLIN assumption of the CGW's compiler is slightly different from our assumption. In particular, it has two fewer group elements in G_2.

Table 2: Efficiency Comparison of Inner Product Encryption (IPE) between encodings.

Scheme	Assumption	PK	SK	CT	Decryption
Wee [32]	SDs	$\ell\|G_N\| + \|G_{N,T}\|$	$2\|G_N\|$	$(\ell+1)\|G_N\| + \|G_{N,T}\|$	$2P + \ell E$
CGW [9]	SXDH	$(2\ell+4)\|G_1\| + \|G_T\|$	$4\|G_2\|$	$2(\ell+1)\|G_1\| + \|G_T\|$	$4P + 2\ell E$
	DLIN	$(6\ell+8)\|G_1\| + 2\|G_T\|$	$6\|G_2\|$	$3(\ell+1)\|G_1\| + \|G_T\|$	$6P + 3\ell E$
Ours	*DLIN*	$(11+\ell)\|G_1\| + \|G_T\|$	$8\|G_2\| + \|\mathbb{Z}_p\|$	$(7+\ell)\|G_1\| + (\ell-1)\|\mathbb{Z}_p\| + \|G_T\|$	$8P + \ell E$

ℓ: the size of a predicate vector (the length of common parameter in the encoding),
P: Pairing computation, E: Exponentiations over a group element,
G_N and $G_{N,T}$: group elements of a composite order N,
G_1, G_2 and G_T: group elements of order p of $e : G_1 \times G_2 \to G_T$

Table 3: Efficiency Comparison of Public Attribute IPE between encodings.

Scheme	Assumption	PK	SK	CT	Decryption
CGW [9]	SXDH	$(2\ell+4)\|G_1\| + \|G_T\|$	$(2\ell+4)\|G_2\|$	$4\|G_1\| + \|G_T\|$	$4P + 2\ell E$
	DLIN	$(6\ell+8)\|G_1\| + 2\|G_T\|$	$(3\ell+6)\|G_2\|$	$6\|G_1\| + \|G_T\|$	$6P + 3\ell E$
Ours, AL [4]	*DLIN*	$(11+\ell)\|G_1\| + \|G_T\|$	$(6+\ell)\|G_2\| + (\ell-1)\|\mathbb{Z}_p\|$	$9\|G_1\| + \|\mathbb{Z}_p\| + \|G_T\|$	$8P + \ell E$

ℓ: the size of a predicate vector (the length of common parameter in the encoding), P: Pairing computation, E: Exponentiation over a group element,
G_1, G_2 and G_T: group elements of order p of $e : G_1 \times G_2 \to G_T$

erarchical Identity Based Encryption with short ciphertexts. Particularly, dual spatial encryption is a new primitive. It is a symmetric conversion of a spatial encryption [17]. In this primitive, an affine space and an affine vector are taken to generate ciphertexts and keys, respectively. Moreover, in the appendix we describe as encodings a number of existing schemes such as IBE [31], (Public Attribute) Inner Product Encryption [4], Spatial Encryption and Doubly Spatial Encryption [8] to show the versatility of our framework.

1.2 Our technique

Our encoding framework generalizes Waters' dual system encryption methodology [31] which is widely used to analyze PE schemes. In Waters' dual system encryption, private keys and ciphertexts are changed into auxiliary types, namely semi-functional keys and semi-functional ciphertexts in the security analysis. After converting all keys and the challenge ciphertext to semi-functional type, proving security becomes much easier in their methodology since semi-functional keys cannot decrypt semi-functional ciphertexts. Prior encodings [2, 32] in composite order groups and their compilers [9, 3, 1] in prime order groups also generalized and utilized the dual system encryption methodology. The most distinctive feature of our encoding compared to theirs is our compiler.

Our compiler is constructed for tag based compiler by utilizing and expanding Waters' IBE [31]. Therefore, our compiler is adaptively secure in prime order groups under the standard DLIN assumption (which is the same as Water's IBE).

The critical part of the dual system encryption is proving semi-functional key invariance. In this proof, it is shown that a normal key and a semi-functional key are indistinguishable when the challenge ciphertext is already fixed as semi-functional. Therefore, the key becomes a valid key into an invalid key against the challenge ciphertext since the semi-functional challenge ciphertext can be decrypted only by a normal key. In Waters' IBE, tags are used to hide the type of the challenge key against not only the adversary but also the simulator. The simulator can try to distinguish the type of the challenge key by generating a valid semi-functional ciphertext to be decrypted only if the key is normal. This trial must be hindered in the analysis. Tags take an important role to restrict the simulator's trial. In Waters' IBE, tags in the challenge key and the challenge ciphertext are enforced to share the same values. In particular, they become $h_1 \cdot ID_{key} + h_2$ and $h_1 \cdot ID_{ct} + h_2$ where h_1 and h_2 are values which are initially information theoretically hidden to the adversary. Therefore, if the simulator generates a ciphertext to test the challenge key, the simulator can only simulate the challenge key with the same tag as the ciphertext, such that the self-decryption cannot be used to distinguish the challenge key because decryption requires two distinct tags. At the same time, since the values of h_1 and h_2 are hidden to the adversary, the correlation between tags in the challenge ciphertext and the challenge key is also hidden since they are pairwise independent. In other words, tags are randomly distributed to the adversary.

In our framework, tags have structures. We reveal the structures of tags, but they take as inputs random values (e.g. h_1 and h_2 in Waters' IBE). In more detail, in our compiler, tags are constructed by the encodings $\mathbf{kE}$ and $\mathbf{cE}$ but take random inputs instead of public parameters. Formally, tags in our compiler are generated as $\mathbf{kE}(x, \mathbf{h}')$ and $\mathbf{cE}(y, \mathbf{h}'')$ where x and y are predicates and $\mathbf{h}'$ and $\mathbf{h}''$ are random values. Therefore, our tags are not random but they retain structures. This approach is actually beneficial for our encoding since we describe tags more formally, but it still works for the dual system encryption methodology. Particularly, in the key invariance proof, those tags must share the same random values (i.e. $\mathbf{h}' = \mathbf{h}''$). This enforces the simulator's trial to fail as in the Waters' IBE system during the decryption process. Also, sharing inputs of encodings can be hidden by utilizing the independence argument such as pairwise independence for IBE. Requiring independence between tags may be a bit more strict than the similar property of the previous encodings. For example, we do not know how linear secret sharing scheme [6] can be utilized into our encoding, but it provides efficiency benefits for PE and still flexible to capture a number of PE schemes.

Duality Another distinct feature of our encodings is that required properties for $\mathbf{kE}$ and $\mathbf{cE}$ are identical. This is useful since without any conversion technique or efficiency loss, one encoding scheme realizes two encryption schemes; one scheme uses $\mathbf{kE}$ for a key and $\mathbf{cE}$ for a ciphertext and the other scheme uses $\mathbf{cE}$ for a key and $\mathbf{kE}$ for a ciphertext. The previous encodings require a new variable incurring efficiency loss for symmetric conversion [5]. We introduce several new schemes as instances of our encoding. Some of them are generated as the symmetric conversions of existing schemes (e.g. Dual spatial encryption as the symmetric conversion of spatial encryption [8]).

2 Related Works

Dual system encryption [31] provides a break-through technique of proving the security of PE. It implements auxiliary types of keys and ciphertexts, namely semi-functional keys and semi-functional ciphertexts, appearing only in the security proof. Subsequently, it shows that a security game consisting of semi-functional keys and semi-functional ciphertexts is indistinguishable from the original security game. Since semi-functional keys cannot decrypt semi-functional ciphertexts, the security proof for the transformed game becomes much easier than that of the original game. Waters showed that dual system encryption is a powerful tool in public key encryptions and signatures by introducing a number of adaptive encryption schemes.

Several encryption systems [8, 13, 4] have been introduced in prime order groups under standard assumption. In particular, all of them share similar constructions and security proofs. Interestingly, their techniques are quite different from those of dual system groups. They are more similar to Waters' IBE [31], but provide different predicates for their own purposes. Compared with similar constructions in composite order groups [32, 19, 4], they are considered to be efficient and secure since they are constructed in prime order groups and their security depends only on standard assumption.

Encoding frameworks [2, 32] well formalize the core properties that the dual system encryption requires. The frameworks consist of syntax and a compiler of encodings. PE schemes were simply written by encoding instances in the frameworks. Then, the compiler is applied to instances of encodings to result in encryption schemes. Those outputs are also adaptively secure since the adaptive security of the compiler is already proved using properties defined in the syntax. Initially, they [2, 32] were suggested only in composite order groups. Several techniques [15, 18, 30, 20, 23] to convert encryption systems in composite order to those in prime order have also been proposed. Nevertheless, the techniques in [15, 18, 30] are not applicable to dual system encryption since they do not hide parameters. It means that it is not applicable to encoding frameworks.

Dual Pairing Vector Spaces (DPVS) [24, 26, 25] have been widely used as a tool that overcomes the inefficiency of composite order groups. In DPVS, core properties which are accomplished by subgroups of composite order groups for adaptive security are featured by orthogonal vectors in prime order groups. DPVS has been used not only to achieve PE schemes directly [21, 24, 25, 27], but also to convert schemes from composite order groups to prime order groups [23, 20, 9]. Lewko and Waters suggest a generic technique in [20] to transform a construction in composite order groups into prime order groups by utilizing DPVS, but it still incurs a loss in efficiency caused by the size of vectors. The technique suggested by [20] requires the size of vectors to increase linearly with a size of predicates when DPVS is used to convert a PE scheme in composite order groups into prime order groups.

Recently, adaptively secure IPE which has a good efficiency was introduced from Ramanna [28]. It is adaptively secure with a short ciphertext in prime order groups. Interestingly, their construction also uses tags as ours although their scheme is not a generic construction as ours. Also, their scheme has shorter fixed parameters in both keys and ciphertexts compared to our general construction, but their scheme relies on the SXDH assumption which is stronger than DLIN assumption in our construction.

Therefore, one may think that their scheme is a trade-off between security and efficiency compared to IPE scheme in our works.

There exist variants of Waters' IBE from Ramanna, Chatterjee and Sarkar [29] and Lewko and Waters [22]. Since our encoding framework generalizes Water's IBE, These variants may be also applicable to our generic construction. Using those variants one may achieve PE schemes which have fewer fixed elements in keys and ciphertexts, but under stronger assumptions as those in Ramanna [28].

3 Background

3.1 Bilinear Maps

We let G_1, G_2 and G_T denote three multiplicative cyclic groups of prime order p. Also, we let g_1 and g_2 be generators of G_1 and G_2, resp., and e be a bilinear map, $e : G_1 \times G_2 \to G_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g_1, g_2) \neq 1$.

We say that G_1 and G_2 are bilinear groups if the group operation in G_1 and G_2 and the bilinear map $e : G_1 \times G_2 \to G_T$ are efficiently computable. If $G_1 \neq G_2$, the map e is an asymmetric bilinear map. Otherwise, we can simply denote G_1 and G_2 as G and call $e : G \times G \to G_T$ a symmetric bilinear map.

3.2 Complexity Assumptions

We expand both the DLIN and the DBDH into asymmetric bilinear maps. Therefore, we let G_1, G_2, and G_T be prime order groups of order p such that $e : G_1 \times G_2 \to G_T$ where e is an asymmetric bilinear map. Moreover, we use subscripts to denote the type of groups. For example, g_1 denotes a generator of G_1, and g_2 denotes a generator of G_2.

(Asymmetric) Decisional Bilinear Diffie-Hellman (*DBDH*) assumption. Let g_1 and g_2 be a generator of G_1 and of G_2, respectively. Let c_1, c_2 and c_3 be selected randomly from $\mathbb{Z}_p$. Given $\{g_1, g_1^{c_2}, g_1^{c_3} \in G_1, g_2, g_2^{c_1}, g_2^{c_2} \in G_2, T \in G_T\}$, there is no PPT algorithm that can distinguish whether T is $e(g_1, g_2)^{c_1 c_2 c_3}$ or a random from G_T with a non-negligible advantage.

(Asymmetric) Decisional Linear (*DLIN*) assumption Let g_1 and g_2 be random generators of G_1 and G_2, respectively. Let y_f, y_ν, c_1, c_2 be selected randomly from $\mathbb{Z}_p$ set $f_1 = g_1^{y_f}, \nu_1 = g_1^{y_\nu}, f_2 = g_2^{y_f}$ and $\nu_2 = g_2^{y_\nu}$. Given $\{g_1, f_1, \nu_1, g_1^{c_1}, f_1^{c_2}, T \in G_1, g_2, f_2, \nu_2 \in G_2\}$, there is no PPT algorithm can distinguish whether T is $\nu_1^{c_1+c_2}$ or a random from G_1 with a non-negligible advantage.

It is worth noting that (Asymmetric) DBDH assumption also reduced to (Asymmetric) DLIN assumption.

Proposition 1. *Suppose that there exists an algorithm $\mathcal{A}$ which breaking* (Asymmetric) DBDH *with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* (Asymmetric) DLIN *assumption with advantage ϵ.*

Proof. $\mathcal{B}$ takes $\{g_1, f_1, \nu_1, g_1^{c_1}, f_1^{c_2}, T, g_2, f_2, \nu_2\}$ as an instance from *(Asymmetric) DLIN* assumption. $\mathcal{B}$ will simulate *(Asymmetric) DBDH* from the instance using $\mathcal{A}$ who breaks *(Asymmetric) DLIN* assumption with non-negligible advantage.

If $\mathcal{A}$ requests a instance of *(Asymmetric) DBDH* $\{\tilde{g}_1, \tilde{g}_1^{\tilde{c}_2}, \tilde{g}_1^{\tilde{c}_3}, \tilde{g}_2, \tilde{g}_2^{\tilde{c}_1}, \tilde{g}_2^{\tilde{c}_2}, \tilde{T}\}$ to break *(Asymmetric) DLIN*, the algorithm sets

$$\tilde{g}_1 = g_1, \tilde{g}_1^{\tilde{c}_2} = f_1, \tilde{g}_1^{\tilde{c}_3} = g_1{}^{\tilde{c}_1}, \tilde{g}_2 = g_2, \tilde{g}_2^{\tilde{c}_1} = \nu_2, \tilde{g}_2^{\tilde{c}_2} = f_2, \tilde{T} = e(T, f_2)/e(f_1^{c_2}, \nu_2).$$

This implicitly sets $\tilde{c}_1 = y_\nu$, $\tilde{c}_2 = y_f$ and $\tilde{c}_3 = c_1$ where y_ν and y_f are the discrete logarithms of ν_1 and f_1 to the base g_1 modulo p, respectively. If T is $\nu_1^{c_1+c_2}$, then $\tilde{T} = e(T, f_2)/e(f_1^{c_2}, \nu_2) = e(\nu_1, f_2)^{c_1} = e(\tilde{g}_1^{\tilde{c}_1}, \tilde{g}_2^{\tilde{c}_2})^{\tilde{c}_3} = e(\tilde{g}_1, \tilde{g}_2)^{\tilde{c}_1\tilde{c}_2\tilde{c}_3}$. Otherwise, if T is a random element from G_1, $\tilde{T}$ is randomized by T.

3.3 Predicate Encryption

PE definition and its adaptive security are adopted from [2, 32].

Definition of Predicate Encryption For a predicate $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, our PE consists of Setup, Encrypt, KeyGen and Decrypt as follows:

Setup $(1^\lambda, \ell) \to (PK, MSK)$: takes as input a security parameter 1^λ and an integer ℓ allocated to a predicate. The output is a public parameter PK and a master secret key MSK.

KeyGen $(x, MSK, PK) \to SK$: takes as input a predicate $x \in \mathcal{X}$, a master secret key MSK and a public parameter PK. The output is a private key SK.

Encrypt $(y, M, PK) \to CT$: takes as input a description $y \in \mathcal{Y}$, a public parameter PK and a plaintext M. The output is a ciphertext CT.

Decrypt $(x, y, SK, CT) \to M$: takes as input a secret key SK for x and a ciphertext CT for y. If $R(x, y) = 1$, the output is M. Otherwise, $\perp$.

Correctness. For all $M, x \in \mathcal{X}, y \in \mathcal{Y}$ such that $R(x, y) = 1$, if SK is the output of KeyGen(x, MSK, PK) and CT is the output of Encrypt(y, M, PK) where PK and MSK are the outputs of Setup($1^\lambda, \ell$), then Decrypt(x, y, SK, CT) outputs M.

Definition of Adaptive Security of Predicate Encryption [2]

With q_t private key queries where q_t is polynomial, a PE scheme for a predicate R is adaptively secure if there is no PPT adversary $\mathcal{A}$ which has a non-negligible advantage in the game between $\mathcal{A}$ and the challenger $\mathcal{C}$ defined below.

Setup**:** The challenger runs Setup($1^\lambda, \ell$) to create (PK, MSK). PK is sent to $\mathcal{A}$.

Phase 1**:** The adversary requests a private key for $x_i \in \mathcal{X}$ for $i \in [1, q_1]$. For each x_i, the challenger returns SK_i created by running KeyGen(x_i, MSK, PK).

Challenge**:** When the adversary requests the challenge ciphertext for $y \in \mathcal{Y}$ such that $R(x_i, y) = 0\ \forall i \in [1, q_1]$, and submits equal-length messages M_0 and M_1, the challenger randomly selects b from $\{0, 1\}$ and returns the challenge ciphertext CT created by running Encrypt(y, M_b, PK).

Phase 2**:** This is identical to Phase 1 except the additional restriction that $x_i \in \mathcal{X}$ for $i \in [q_1 + 1, q_t]$ such that $R(x_i, y) = 0; \forall i \in [q_1 + 1, q_t]$.

Guess: The adversary outputs $b' \in \{0, 1\}$. If $b = b'$, then the adversary wins.

We define the advantage of the adversary against a predicateencryption as

$$Adv_{\mathcal{A}}^{PE}(\lambda) := |\Pr[b = b'] - 1/2|.$$

3.4 Notations

Throughout the paper, we use bold font to denote vectors. Furthermore, vector exponentiations of group elements imply vector group elements. For example, we let $\mathbf{a} = (a_1, a_2)$ where $\mathbf{a} = (a_1, a_2) \in \mathbb{Z}_p^2$. For a group element g, $g^{\mathbf{a}}$ is equal to (g^{a_1}, g^{a_2}). In addition, multiplication of vectors in exponents implies component-wise product of two vectors. For example, $g^{\mathbf{ab}}$ is equal to $(g^{a_1 b_1}, g^{a_2 b_2})$ where $\mathbf{b} = (b_1, b_2) \in \mathbb{Z}_p^2$. Similarly, a scalar exponentiation to a vector of group elements means a scalar multiplication to a vector in exponent. For example, $(g^{(a_1,a_2)})^r = (g^{(ra_1,ra_2)})$ where $r \in \mathbb{Z}_p$. Also, a multiplication of vector groups implies an addition of vectors in their exponents (e.g. $g^{\mathbf{a}} g^{\mathbf{b}} = g^{\mathbf{a}+\mathbf{b}}$). It should be noted that this multiplication is possible only if $|\mathbf{a}| = |\mathbf{b}|$. When it comes to a pairing operation, a pairing with vectors implies multiple pairing computations, that is, $e(g, g^{\mathbf{a}})$ requires two pairing computations $e(g, g^{a_1})e(g, g^{a_2})$ where $\mathbf{a} = (a_1, a_2) \in \mathbb{Z}_p^2$, but the same result is achieved only by one pairing since $e(g, g^{a_1} g^{a_2}) = e(g, g^{a_1})e(g, g^{a_2})$.

4 Tag based encoding

For a predicate $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, tag based encoding $\mathbf{TE}(R)$ is a tuple of (ℓ, $\mathbf{kE}$, $\mathbf{cE}$). In an encoding (ℓ, $\mathbf{kE}$, $\mathbf{cE}$), ℓ is an integer allocated for a predicate R (e.g. the size of a universe of attributes in ABE, the dimension of an affine space in spatial encryption) and used to generate common parameter $\mathbf{h} \in \mathbb{Z}_p^\ell$. Also, $\mathbf{kE}(x, \mathbf{h})$ and $\mathbf{cE}(y, \mathbf{h})$ are two deterministic algorithms which take as inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, resp. together with $\mathbf{h}$.

We let ℓ_k and ℓ_c denote the sizes of $\mathbf{kE}(x, \mathbf{h})$ and $\mathbf{cE}(y, \mathbf{h})$ (i.e. $\ell_k = |\mathbf{kE}(x, \mathbf{h})|$ and $\ell_c = |\mathbf{cE}(y, \mathbf{h}')|$), resp. Then, tag based encodings satisfy following properties:

Property 1. (Reconstruction) . For all (x, y) such that $R(x, y) = 1$, there exists an efficient algorithm to compute non-zero vectors $\mathbf{m}_x \in \mathbb{Z}_p^{\ell_k}$ and $\mathbf{m}_y \in \mathbb{Z}_p^{\ell_c}$ such that

$$\mathbf{m}_x \mathbf{kE}(x, \mathbf{h}) = \mathbf{m}_y \mathbf{cE}(y, \mathbf{h}), \quad \forall \mathbf{h} \in \mathbb{Z}_p^\ell.$$

Property 2. (*Linearity*) For all $(x, y, \mathbf{h}', \mathbf{h}'') \in \mathcal{X} \times \mathcal{Y} \times \mathbb{Z}_p^\ell \times \mathbb{Z}_p^\ell$,

$$\mathbf{kE}(x, \mathbf{h}') + \mathbf{kE}(x, \mathbf{h}'') = \mathbf{kE}(x, \mathbf{h}' + \mathbf{h}'') \text{ and}$$

$$\mathbf{cE}(y, \mathbf{h}') + \mathbf{cE}(y, \mathbf{h}'') = \mathbf{cE}(y, \mathbf{h}' + \mathbf{h}'').$$

Property 3. (h-*hiding*) For all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R(x, y) = 0$,

$$(x, y, \mathbf{kE}(x, \mathbf{h}), \mathbf{cE}(y, \mathbf{h})) \text{ and } (x, y, \mathbf{kE}(x, \mathbf{h}), \mathbf{cE}(y, \mathbf{h}'))$$

are statistically indistinguishable where $\mathbf{h}$ and $\mathbf{h}'$ are randomly selected from $\mathbb{Z}_p^\ell$,

Remark 1. *Reconstruction* is necessary for the correctness of our construction. In our construction, $\mathbf{kE}(x, \mathbf{h})$ and $\mathbf{cE}(y, \mathbf{h})$ cancel each other out. Hence, the property implies that there exists an efficient algorithm to make both tuples identical.

An example of tag based encodings. We provide a simple IBE scheme as an instance of our encoding from Waters' IBE [31]. This encoding results in an adaptively secure IBE scheme via our compiler introduced in the next section.

Let $\mathcal{X} = \mathcal{Y} := \mathbb{Z}_p$. For all $ID \in \mathcal{X}$ and $ID' \in \mathcal{Y}$, $R(ID, ID') = 1$ iff $ID = ID'$.

- $\ell = 2$ and $\mathbf{h} = (y_u, y_h) \in \mathbb{Z}_p^2$
- $\mathbf{kE}(ID, (y_u, y_h)) := (y_u ID + y_h) \in \mathbb{Z}_p$
- $\mathbf{cE}(ID', (y_u, y_h)) := (y_u ID' + y_h) \in \mathbb{Z}_p$
- **Reconstruction:** This is an exact cancellation. Therefore, $\mathbf{m}_x = \mathbf{m}_y = 1$.
- **Linearity:** For all $\mathbf{h}' = (y'_u, y'_h)$,

$$\begin{aligned}\mathbf{kE}(ID, (y'_u, y'_h)) + \mathbf{kE}(ID, (\tilde{y}_u, \tilde{y}_h)) &= y'_u ID + y'_h + \tilde{y}_u ID + \tilde{y}_h \\ &= \mathbf{kE}(ID, (y'_u + \tilde{y}_u, y'_h + \tilde{y}_h))\end{aligned}$$

  The linearity of $\mathbf{cE}(ID', (y'_u, y'_h))$ is identical showed with $\mathbf{kE}(ID, (y'_u, y'_h))$.
- ***h*-hiding:** Given an instance (ID, ID', $\mathbf{kE}(ID, (y_u, y_h))$, $\mathbf{cE}(ID', (y_u, y_h))$), because $\mathbf{kE}$ and $\mathbf{cE}$ are pairwise independence functions and the values of y_u and y_h are hidden. If $ID \neq ID'$, they do not correlate to each other. Therefore, sharing y_u and y_h between $\mathbf{kE}$ and $\mathbf{cE}$ are statistically hidden.

5 Our Compiler

Our compiler is similar to those of Waters' IBE [31]. The main differences between Waters' IBE and ours are the way of generating tags in KeyGen and Encrypt and the types of bilinear maps which are equipped with. In particular, tags in our construction have structures although tags of Waters' IBE are created randomly.

5.1 The Construction

For a tag based encoding $\mathbf{TE}(R)$ for a predicate R where $R : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$, with ℓ which is an integer to associated with R, $\mathbf{PE}_A(\mathbf{TE}(R))$ is constructed as follows.

- Setup(1^λ, ℓ): The algorithm takes ℓ of the encoding as an input. Then, it randomly generates three groups G_1, G_2 and G_T from $\mathcal{G}(\lambda, p)$. Next, it generates $g_1 \in G_1$ and $g_2 \in G_2$ and exponents α, y_u, y_v, y'_v, y_w, a_1, a_2, b, h_1, ..., $h_\ell \in \mathbb{Z}_p$. Let $\tau_1 = g_1^{y_u + a_1 \cdot y_v}, \tau'_1 = g_1^{y_u + a_2 \cdot y'_v}$ and $\mathbf{h} = (h_1, ..., h_\ell)$. The MSK consists of $(g_2, g_2^\alpha, g_2^{\alpha \cdot a_1}, g_2^b, u_2 = g_2^{y_u}, v_2 = g_2^{y_v}, v'_2 = g_2^{y'_v}, w_2 = g_2^{y_w}, g_2^{\mathbf{h}})$. It publishes the public parameters PK as follows

$$g_1, g_1^b, g_1^{a_1}, g_1^{a_2}, g_1^{b \cdot a_1}, g_1^{b \cdot a_2}, \tau_1, \tau'_1, \tau_1^b, {\tau'_1}^b, w_1 = g_1^{y_w}, g_1^{\mathbf{h}}, e(g_1, g_2)^{\alpha \cdot a_1 \cdot b}$$

- Keygen(MSK, PK, x): The algorithm chooses randomly $r_1, r_2, z_1, z_2, h'_1, ..., h'_\ell \in \mathbb{Z}_p$ and sets $r = r_1 + r_2$ and $\mathbf{Tag}_k = \mathbf{kE}(x, \mathbf{h}')$ where $\mathbf{h}'$ is equal to $(h'_1, ..., h'_\ell)$. Then, it sets

$$D_1 = g_2^{\alpha \cdot a_1} u_2^r, D_2 = g_2^{-\alpha} v_2^r g_2^{z_1}, D_3 = (g_2^b)^{-z_1}, D_4 = {v'_2}^r g_2^{z_2}, D_5 = (g_2^b)^{-z_2},$$

$$D_6 = (g_2^b)^{r_2}, D_7 = g_2^{r_1}, \mathbf{K} = (g_2^{\mathbf{kE}(x, \mathbf{h})} w_2^{\mathbf{Tag}_k})^{r_1}.$$

  It outputs $SK = (D_1, ..., D_7, \mathbf{K}, \mathbf{Tag}_k)$.

- Encrypt(PK, M, y): The algorithm randomly selects s_1, s_2, t, $h''_1, ..., h''_\ell \in \mathbb{Z}_p$ and set $s = s_1 + s_2$, and $\mathbf{Tag}_c = \mathbf{cE}(y, \mathbf{h}'')$ where $\mathbf{h}''$ is equal to $(h''_1, ..., h''_\ell)$. It sets

$$C = M \cdot (e(g_1, g_2)^{\alpha a_1 \cdot b})^{s_2},\ C_1 = (g_1^b)^s,\ C_2 = (g_1^{b \cdot a_1})^{s_1},\ C_3 = (g_1^{a_1})^{s_1},$$

$$C_4 = (g_1^{b \cdot a_2})^{s_2},\ C_5 = (g_1^{a_2})^{s_2}, C_6 = \tau_1^{s_1} {\tau'_1}^{s_2}, C_7 = (\tau_1^b)^{s_1} ({\tau'_1}^b)^{s_2} w_1^{-t}, C_8 = g_1^t,$$

$$\mathbf{E} = (g_1^{\mathbf{cE}(y,\mathbf{h})} w_1^{\mathbf{Tag}_c})^t.$$

  It outputs $CT = (C, C_1, ..., C_8, \mathbf{E}, \mathbf{Tag}_c)$[5].
- Decrypt(x, y, SK, CT, PK): First, the algorithm calculates

$$A_1 = e(C_1, D_1)e(C_2, D_2)e(C_3, D_3)e(C_4, D_4)e(C_5, D_5),$$

$$A_2 = e(C_6, D_6)e(C_7, D_7).$$

  Since $R(x, y) = 1$, there exist reconstruction vectors $\mathbf{m}_x$ and $\mathbf{m}_y$ s.t. $\mathbf{m}_x \mathbf{kE}(x, \mathbf{h}) = \mathbf{m}_y \mathbf{cE}(y, \mathbf{h})$ (by Property 1). If $\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c$ is 0, it aborts. Otherwise,

$$A_3 = e(C_8, \mathbf{K}^{\mathbf{m}_x})/e(\mathbf{E}^{\mathbf{m}_y}, D_7) = e(g_1, g_2)^{y_w r_1 t (\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)}.$$

  Therefore, $M = C \cdot A_2/(A_1 \cdot A_3^{1/(\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)})$.

Correctness. Calculating A_1/A_2 is trivial and can be found in [31]. We only point out that $A_1/A_2 = e(g_1, g_2)^{\alpha a_1 \cdot b s_2} e(g_1, w_2)^{-r_1 t}$. For $\mathbf{m}_x$ and $\mathbf{m}_y$ such that $\mathbf{m}_x \mathbf{kE}(x, \mathbf{h}) = \mathbf{m}_y \mathbf{cE}(y, \mathbf{h})$, the correctness of A_3 is calculated as follows

$$\begin{aligned} A_3 &= \frac{e(C_8, \mathbf{K}^{\mathbf{m}_x})}{e(\mathbf{E}^{\mathbf{m}_y}, D_7)} = \frac{e(g_1^t, (g_2^{\mathbf{kE}(x,\mathbf{h})} w_2^{\mathbf{Tag}_k})^{r_1 \cdot \mathbf{m}_x})}{e((g_1^{\mathbf{cE}(y,\mathbf{h})} w_1^{\mathbf{Tag}_c})^{t \cdot \mathbf{m}_y}, g_2^{r_1})} \\ &= \frac{e(g_1, g_2)^{r_1 \cdot t \cdot \mathbf{m}_x \mathbf{kE}(x,\mathbf{h})} e(g_1, w_2)^{r_1 \cdot t \cdot \mathbf{m}_x \mathbf{Tag}_k}}{e(g_1, g_2)^{r_1 \cdot t \cdot \mathbf{m}_y \cdot \mathbf{cE}(y,\mathbf{h})} e(g_1, w_2)^{r_1 \cdot t \cdot \mathbf{m}_y \cdot \mathbf{Tag}_c}} \\ &= e(g_1, w_2)^{r_1 t \cdot (\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)}. \end{aligned}$$

Therefore, $M = C \cdot A_2/(A_1 \cdot A_3^{1/(\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)})$.

Remark 2. Alternatively, to reduce the number of pairing computations, we sets $\mathbf{m'}_x = \mathbf{m}_x/(\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)$ and $\mathbf{m}'_y = \mathbf{m}_y/(\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)$.

Then, the decryption can be done by calculating

$$A'_1 := e(C_1, D_1)e(C_2, D_2)e(C_3, D_3)e(C_4, D_4)e(C_5, D_5)/e(C_6, D_6),$$

$$A'_2 := e(C_8, \mathbf{K}^{\mathbf{m}'_x/a_1})/e(\tilde{\mathbf{E}}, D_7).$$

where $\tilde{\mathbf{E}} := (C_7, \mathbf{E}^{\mathbf{m}'_y})$. Finally, M is retrieved since $M = C/(A'_1 \cdot A'_2)$.

Theorem 1. *Suppose there exists a tag based encoding* **TE***, then our* $\mathbf{PE}_A$**(TE)** *is adaptively secure under the* (Asymmetric) DLIN *assumption.*

Proof. This is proved by lemmas 1 to 3.

[5] Linearity of $\mathbf{kE}(x, \mathbf{h})$ and $\mathbf{cE}(x, \mathbf{h})$ implies that $\mathbf{kE}$ and $\mathbf{cE}$ are linear functions over $\mathbf{h}$ when x and y are given. Therefore, $g^{\mathbf{kE}(x,\mathbf{h})}$ and $g^{\mathbf{cE}(y,\mathbf{h})}$ can be efficiently computed from $g^{\mathbf{h}}$ if x and y are given.

6 Security Analysis

Semi-functional Ciphertext. By running Encrypt algorithm for a message M and an input y, the algorithm generates a normal ciphertext $CT = (C', C'_1, ..., C'_8, \mathbf{E}', \mathbf{Tag}'_c)$. Then, it randomly selects $\kappa \in \mathbb{Z}_p$ and sets

$$C = C', \quad C_1 = C'_1, \quad C_2 = C'_2, \quad C_3 = C'_3, \quad C_4 = C'_4 g_1^{b a_2 \kappa},$$

$$C_5 = C'_5 g_1^{a_2 \kappa}, \; C_6 = C'_6 {v'_1}^{a_2 \kappa}, \; C_7 = C'_7 {v'_1}^{a_2 b \kappa}, \; C_8 = C'_8, \; \mathbf{E} = \mathbf{E}', \; \mathbf{Tag}_c = \mathbf{Tag}'_c.$$

Semi-functional Key. The algorithm generates a normal key $SK = (D'_1, \ldots, D'_7, \mathbf{K}', \mathbf{Tag}'_k)$ by running Keygen algorithm for an input x. Then, it sets

$$D_1 = D'_1 g_2^{-a_1 a_2 \gamma}, \quad D_2 = D'_2 g_2^{a_2 \gamma}, \quad D_3 = D'_3, \quad D_4 = D'_4 g_2^{a_1 \gamma}$$

$$D_5 = D'_5, \quad D_6 = D'_6, \quad D_7 = D'_7, \quad \mathbf{K} = \mathbf{K}', \quad \mathbf{Tag}_k = \mathbf{Tag}'_k.$$

It should be noted that $e(g_1, g_2)^{a_1 a_2 b \kappa \gamma}$ will be added to the message to be encrypted if the semi-functional key is used to decrypt the semi-functional ciphertext.

Security Games

Game_{real}**:** This game is a real game. It is identical to the adaptive security model.

Game_i**:** This game is identical to Game_{real} except the challenge ciphertext and the first i keys. In this game, the challenge ciphertext and the first i keys are semi-functional.

Game_{final}**:** This game is identical to Game_{q_t} except the challenge ciphertext where q_t is the total number of key queries. In this game, the challenge ciphertext is still semi-functional, but it is an encryption of a random message.

First, we prove that Game_{real} and Game_0 are indistinguishable (*semi-functional ciphertext invariance*) in lemma 1. Then, we show that Game_{k-1} is also indistinguishable from Game_k (*semi-functional key invariance*) in lemma 2. Finally, in lemma 3, we prove the invariance between Game_{q_t} and Game_{final} (*semi-functional security*). This completes the security analysis since no attacker has a non-negligible advantage in Game_{final}.

Lemma 1. (Semi-functional Ciphertext Invariance) *Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes Game_{real} and Game_0 with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* (Asymmetric) DLIN *assumption with advantage ϵ.*

Proof. Firstly, $\mathcal{B}$ takes an $\{g_1, f_1, \nu_1, g_1^{c_1}, f_1^{c_2}, T, g_2, f_2, \nu_2\}$ as an instance from (Asymmetric) *DLIN* assumption. Depending on the value of T, $\mathcal{B}$ will simulate Game_{real} or Game_0 to take an advantage from $\mathcal{A}$ which can distinguish both games with non-negligible advantage ϵ.

Setup The algorithm, $\mathcal{B}$, chooses exponents $\alpha, b, y_u, y_v, y'_v, y_w, h_1, ...h_\ell$ randomly from $\mathbb{Z}_p$. It sets $u_1 = g_1^{y_u}$, $v_1 = g_1^{y_v}$ and $v'_1 = g_1^{y'_v}$. Then, it publishes public parameters as:

$$\text{PK} := \{g_1, \, g_1^b, \, g_1^{a_1} = f_1, \, g_1^{a_2} = \nu_1, \, g_1^{b \cdot a_1} = f_1^b, \, g_1^{b \cdot a_2} = \nu_1^b, \, w_1 = g_1^{y_w}, g_1^{h_1}, ..., g_1^{h_\ell},$$

$$\tau_1 = g_1^{y_u} f_1^{y_v},\ \tau_1' = g_1^{y_u} \nu_1^{y_v'},\ \tau_1^b,\ {\tau_1'}^b,\ e(g_1,g_2)^{\alpha b a_1} = e(g_1,f_2)^{\alpha b}\}.$$

This implicitly sets $a_1 = y_f$ and $a_2 = y_\nu$ where y_f and y_ν are the discrete logarithms of ν_1 and f_1 to the base g_1 modulo p, respectively. It sets $\mathrm{MSK} := \{g_2, g_2^\alpha, g_2^{\alpha \cdot a_1} = f_2^\alpha, g_2^b, u_2 = g_2^{y_u}, v_2 = g_2^{y_v}, v_2' = g_2^{y_v'}, w_2 = g_2^{y_w}, g_2^{h_1}, ..., g_2^{h_\ell}\}$.

Phase I and II Because $\mathcal{B}$ knows the public parameters and the master secret key, it can generate normal keys using Keygen.

Challenge Ciphertext When the adversary requests the challenge ciphertext for y^* with messages M_0, M_1, $\mathcal{B}$ randomly selects β from $\{0,1\}$. $\mathcal{B}$ runs the encryption algorithm to generate a normal ciphertext $CT' = (C', C_1', ..., C_8', \mathbf{E}', \mathbf{Tag}_c')$ for y^* and M_β. We denote the random exponents of CT' as s_1', s_2', t'. Then it sets the challenge CT as:

$$C = C' \cdot (e(g_1^{c_1}, f_2) \cdot e(g_1, f_2^{c_2}))^{b \cdot \alpha}, C_1 = C_1' \cdot (g_1^{c_1})^b, C_2 = C_2' \cdot (f_1^{c_2})^{-b}, C_3 = C_3' f_1^{c_2},$$

$$C_4 = C_4'(T)^b, \quad C_5 = C_5' \cdot T, \quad C_6 = C_6' \cdot (g_1^{c_1})^{y_u} \cdot (f_1^{c_2})^{-y_v} \cdot T^{y_v'},$$

$$C_7 = C_7' \cdot ((g_1^{c_1})^{y_u} \cdot (f_1^{c_2})^{-y_v} \cdot T^{y_v'})^b, \quad C_8 = C_8', \quad \mathbf{E} = \mathbf{E}', \quad \mathbf{Tag}_c = \mathbf{Tag}_c'.$$

This implicitly sets $s_1 = -c_2 + s_1', s_2 = s_2' + c_1 + c_2$ and $s = s_1 + s_2 = c_1 + s_1' + s_2'$. If $T = \nu_1^{c_1 + c_2}$, the challenge ciphertext is a normal ciphertext, and $Game_{real}$ has been simulated properly. Otherwise, if T is a random, and is denoted as $\nu_1^{c_1+c_2}\nu_1^\kappa$, this is a properly distributed semi-functional key, and $Game_0$ has been simulated properly.

Lemma 2. (Semi-functional Key Invariance) *Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes* Game_{k-1} *and* Game_k *with a non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* (Asymmetric) DLIN *assumption with ϵ.*

Proof. G_1 and G_2 of (Asymmetric) DLIN are reversed. Therefore, $\mathcal{B}$ takes $\{g_1, f_1, \nu_1, g_2, f_2, \nu_2, g_2^{c_1}, f_2^{c_2}, T\}$ as an instance from *(Asymmetric) DLIN* assumption. Depending on the value of T, $\mathcal{B}$ will simulate Game_{k-1} or Game_k to take an advantage from $\mathcal{A}$ which can distinguish. It should be noted that T is in G_2 in the reversed assumption.

Setup $\mathcal{B}$ chooses exponents $\alpha, a_1, a_2, y_v, y_v', y_w, h_1', ...h_\ell', \tilde{h}_1, ...\tilde{h}_\ell$, randomly from $\mathbb{Z}_p$. It sets

$$e(g_1,g_2)^{\alpha \cdot a_1 b} = e(f_1,g_2)^{\alpha \cdot a_1}, g_1 = g_1, g_1^b = f_1, g_1^{b \cdot a_1} = f_1^{a_1}, g_1^{b \cdot a_2} = f_1^{a_2},$$

$$u_1 = \nu_1^{-a_1 a_2}, v_1 = \nu_1^{a_2} \cdot g_1^{y_v}, v_1' = \nu_1^{a_1} \cdot g_1^{y_v'}, \tau_1 = u_1 v_1^{a_1} = g_1^{y_v a_1}, \tau_1' = u_1 v_1'^{a_2} = g_1^{y_v' a_2},$$

$$\tau_1^b = f_1^{y_{v_1} a_1}, {\tau_1'}^b = f_1^{y_v' a_2}, w_1 = f_1 g_1^{y_w}, \{g_1^{h_i} = f_1^{-h_i'} g_1^{\tilde{h}_i}; \forall i \in [1,\ell]\}$$

Then, it publishes the public parameters

$$g_1, g_1^b, g_1^{a_1}, g_1^{a_2}, g_1^{b \cdot a_1}, g_1^{b \cdot a_2}, \tau_1, \tau_1', \tau_1^b, {\tau_1'}^b, w_1, g_1^{\mathbf{h}}, e(g_1,g_2)^{\alpha \cdot a_1 \cdot b}$$

where $\mathbf{h} = (h_1, ..., h_\ell)$. $\mathcal{B}$ sets $\mathrm{MSK} = \{g_2, g_2^\alpha, g_2^{\alpha \cdot a_1}, g_2^b = f_2, u_2 = \nu_2^{-a_1 a_2}, v_2 = \nu_2^{a_2} g_2^{y_v}, v_2' = \nu_2^{a_1} g_2^{y_v'}, w_2 = f_2 g_2^{y_w}, \{g_2^{h_i} = f_2^{-h_i'} g_2^{\tilde{h}_i}; \forall i \in [1,\ell]\}\}$.

In the setting, $\mathbf{h}$ is implicitly set by $\tilde{\mathbf{h}} - y_f \mathbf{h}'$ where $\mathbf{h}' = (h_1', ...h_\ell')$ and $\tilde{\mathbf{h}} = (\tilde{h}_1, ...\tilde{h}_\ell)$ if we write $f_1 = g_1^{y_f}$. $\mathcal{B}$ calculates $g_1^{\mathbf{h}}$ because it knows $g_1, f_1, \tilde{\mathbf{h}}, \mathbf{h}'$. It

should be noted that the values of $\{h'_i; \forall i \in [1, \ell]\}$ are not revealed. It means that they are initially information theoretically hidden because, for all $i \in [1, \ell]$, $\tilde{h}_i$ is uniquely added where h'_i appears.

Phase I and II For the first $k-1$ semi-functional keys, $\mathcal{B}$ generates a normal key and selects γ randomly from $\mathbb{Z}_p$. It then adds semi-functional parts to the normal key . This is possible because $\mathcal{B}$ knows a_1, a_2 and MSK. Similarly, for the rest keys except k^{th} key $(i > k)$, $\mathcal{B}$ can generate normal keys using the key generation algorithm, **KeyGen**, for the same reason.

For the k^{th} key, $\mathcal{B}$ sets $\mathbf{Tag}'_k = \mathbf{kE}(x, \mathbf{h}')$. Then, with $\mathbf{Tag}'_k$, it generates a normal key $SK' = (D'_1, ..., D'_7, \mathbf{K}', \mathbf{Tag}'_k)$ using the key generation algorithm. Then, it reuses $\mathbf{Tag}'_k$ in the k^{th} key (i.e. $\mathbf{Tag}_k = \mathbf{Tag}'_k$) and sets the other elements as follows

$$D_1 = D'_1 T^{-a_1 a_2},\ D_2 = D'_2 T^{a_2} (g_2^{c_1})^{y_v},\ D_3 = D'_3 (f_2^{c_2})^{y_v},\ D_4 = D'_4 T^{a_1} (g_2^{c_1})^{y'_v},$$

$$D_5 = D'_5 (f_2^{c_2})^{y'_v}, D_6 = D'_6 f_2^{c_2}, D_7 = D'_7 (g_2^{c_1}), \mathbf{K} = \mathbf{K}' (g_2^{c_1})^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')}, .$$

We let r'_1, r'_2, z'_1, z'_2 denote the random exponents of SK'. Then, it implicitly sets $z_1 = z'_1 - y_v c_2$ and $z_2 = z'_2 - y'_v c_2$. Also, by *linearity* property,

$$g_2^{\mathbf{kE}(x, \mathbf{h})} = g_2^{\mathbf{kE}(x, -y_f \mathbf{h}' + \tilde{\mathbf{h}})} = g_2^{\mathbf{kE}(x, -y_f \mathbf{h}')} g_2^{\mathbf{kE}(x, \tilde{\mathbf{h}})} = f_2^{-\mathbf{kE}(x, \mathbf{h}')} g_2^{\mathbf{kE}(x, \tilde{\mathbf{h}})}$$

Therefore, the value of $\mathbf{K}'$ can be represented as follows:

$$\mathbf{K}' = (f_2^{-\mathbf{kE}(x, \mathbf{h}')} g_2^{\mathbf{kE}(x, \tilde{\mathbf{h}})} (f_2 g_2^{y_w})^{\mathbf{kE}(x, \mathbf{h}')})^{r'_1} = (g_2^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{r'_1}.$$

This implies that $\mathbf{K} = \mathbf{K}' (g_2^{c_1})^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')} = (g_2^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{r'_1 + c_1}$.

If T is equal to $\nu_2^{c_1 + c_2}$, then the k^{th} key is a normal key with $r_1 = r'_1 + c_1$ and $r_2 = r'_2 + c_2$. Otherwise, if T is $\nu_2^{c_1 + c_2} g_2^{\gamma}$, which means a random group element, then, the k^{th} key is a properly distributed semi-functional key.

Challenge Ciphertext When the adversary requests the challenge ciphertext for y^* with messages M_0, M_1, $\mathcal{B}$ randomly selects β from $\{0, 1\}$. With $\mathbf{Tag}'_c = \mathbf{cE}(y^*, \mathbf{h}')$, $\mathcal{B}$ runs the encryption algorithm to generate a normal ciphertext $CT' = (C', C'_1, ..., C'_8, \mathbf{E}', \mathbf{Tag}'_c)$ for y^* and M_β. We let s'_1, s'_2, t' denote the random exponents of CT'. To make the semi-functional challenge ciphertext, it randomly selects $\kappa \in \mathbb{Z}_p$ and sets $C = C', C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$. Additionally, it sets

$$C_4 = C'_4 f_1^{a_2 \cdot \kappa}, \quad C_5 = C'_5 \cdot g_1^{a_2 \cdot \kappa}, \quad C_6 = C'_6 \cdot {v'_1}^{a_2 \kappa}, C_7 = C'_7 \cdot f_1^{y'_v \cdot \kappa \cdot a_2} \nu_1^{-a_1 \cdot \kappa \cdot y_w \cdot a_2}$$

$$C_8 = g_1^{t'} \cdot \nu_1^{a_1 a_2 \kappa}, \quad \mathbf{E} = \mathbf{E}' \cdot (\nu_1^{\mathbf{cE}(y^*, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{a_1 a_2 \kappa}, \quad \mathbf{Tag}_c = \mathbf{Tag}'_c$$

This implicitly sets $g_1^t = g_1^{t'} \cdot \nu_1^{a_1 a_2 \kappa}$. Also, ${\nu_1}^{a_1 a_2 b \kappa}$ of ${v'_1}^{a_2 b \kappa}$ is cancelled out by w_1^{-t} in C_7.

The fact that $\mathbf{Tag}_c$ and $\mathbf{Tag}_k$ share the same vector $\mathbf{h}'$ is hidden to the adversary by *h-hiding* property since $R(x, y^*) = 0$. Therefore, $\mathbf{Tag}_c$ with correlated $\mathbf{h}'$ can be switched to $\mathbf{Tag}_c$ with a random vector from $\mathbb{Z}^\ell$. Also, $\mathbf{E}$ is valid since

$$\mathbf{E}' = (f_1^{-\mathbf{cE}(y^*, \mathbf{h}')} g_1^{\mathbf{cE}(y^*, \tilde{\mathbf{h}})} (f_1 g_1^{y_w})^{\mathbf{cE}(y^*, \mathbf{h}')})^{t'} = (g_1^{\mathbf{cE}(y^*, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{t'}.$$

The second equality of the above equation holds by *linearity* property.

$\mathcal{B}$ cannot test whether the k^{th} key is normal or semi-functional by creating a ciphertext which can be decrypted only by a normal key because $\mathbf{Tag}_k$ and $\mathbf{Tag}_c$ share $\mathbf{h}'$. It means that $\mathbf{m}_x\mathbf{Tag}_k - \mathbf{m}_{y^*}\mathbf{Tag}_c$ is equal to 0 if the simulator creates a semi-functional ciphertext such that $R(x, y) = 1$. Hence, the decryption algorithm will abort.

Lemma 3. (Semi-functional Security) *Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes Game_{q_t} and Game_{final} with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* (Asymmetric) DBDH *assumption with advantage ϵ.*

Proof. $\mathcal{B}$ takes an $\{g_1, g_1^{c_2}, g_1^{c_3}, g_2, g_2^{c_1}, g_2^{c_2}, T\}$ as an instance from *(Asymmetric) DBDH* assumption. Depending on the value of T, $\mathcal{B}$ will simulate Game_{q_t} or Game_{final} to take an advantage from $\mathcal{A}$ which can distinguish both games.

Setup $\mathcal{B}$ chooses exponents $b, a_1, y_u, y_v, y'_v, y_w, h_1, ...h_\ell$, randomly from $\mathbb{Z}_p$, and sets

$$g_1,\ g_1^b,\ g_1^{a_1}, g_1^{a_2} = g_1^{c_2},\ g_1^{b\cdot a_1},\ g_1^{b\cdot a_2} = (g_1^{c_2})^b, w_1 = g_1^{y_w}, \{g_1^{h_i};\ \forall i \in [1, \ell]\}$$

$$\tau_1 = g_1^{y_u + y_v a_1}, \tau'_1 = g_1^{y_u}(g_1^{c_2})^{y'_v}, \tau_1^b, {\tau'_1}^b, e(g_1, g_2)^{\alpha\cdot a_1 b} = e(g_1^{c_2}, g_2^{c_1})^{ba_1}.$$

It implicitly sets $u_1 = g^{y_u}, v_1 = g^{y_v}, v'_1 = g^{y'_v}, \alpha = c_1 \cdot c_2, a_2 = c_2$. It should be noted that MSK cannot be explicitly calculated because $\mathcal{B}$ does not know g_1^α.

Phase I and II For semi-functional keys, if the adversary requests a private key for x, it randomly generates r_1, r_2, z_1, z_2, γ', $h'_1, ..., h'_\ell$ from $\mathbb{Z}_p$ and sets $\mathbf{Tag}_k = \mathbf{kE}(x, \mathbf{h}')$ where $\mathbf{h}' = (h'_1, .., h'_\ell)$. It, then, creates semi-functional key as follows:

$$D_1 = (g_2^{c_2})^{-\gamma' a_1} u_2^r,\ D_2 = (g_2^{c_2})^{\gamma'} v_2^r g_2^{z_1},\ D_3 = (g_2^b)^{-z_1},\ D_4 = (g_2^{c_1})^{a_1} g_2^{a_1\cdot\gamma'} {v'_2}^r g_2^{z_2},$$

$$D_5 = (g_2^b)^{-z_2}, \quad D_6 = g_2^{r_2\cdot b}, \quad D_7 = g_2^{r_1}, \quad \mathbf{K} = (g_2^{\mathbf{kE}(x,\mathbf{h})} w_2^{\mathbf{Tag}_k})^{r_1}, \quad \mathbf{Tag}_k.$$

where $\mathbf{h} = (h_1, .., h_\ell)$. This implicitly sets $\gamma = c_1 + \gamma'$ and $r = r_1 + r_2$.

Challenge Ciphertext When the adversary requests the challenge ciphertext for y^* with messages M_0, M_1, $\mathcal{B}$ randomly selects β from $\{0, 1\}$. Then it generates random values $\kappa', s_1, t, h''_1, ...h''_\ell$ from $\mathbb{Z}_p$ and sets $\mathbf{Tag}_c = \mathbf{cE}(y^*, \mathbf{h}'')$ where $\mathbf{h}'' = (h''_1, ...h''_\ell)$. It, then, creates the challenge ciphertext as follows:

$$C = M_\beta T^{a_1 b}, C_1 = g_1^{s_1\cdot b} + (g_1^{c_3})^b, C_2 = g_1^{b\cdot a_1\cdot s_1}, C_3 = g_1^{a_1\cdot s_1}, C_4 = (g_1^{c_2})^{\kappa'\cdot b},$$

$$C_5 = (g_1^{c_2})^{\kappa'}, C_6 = \tau_1^{s_1}(g_1^{c_3})^{y_u}(g_1^{c_2})^{y'_v\cdot\kappa'}, C_7 = (\tau_1^b)^{s_1}(g_1^{c_3})^{y_u\cdot b}(g_1^{c_2})^{y'_v\cdot\kappa'\cdot b} w_1^{-t},$$

$$C_8 = g_1^t,\ \mathbf{E} = g_1^{\mathbf{cE}(y^*,\mathbf{h}'')} w_1^{\mathbf{Tag}_c}, \mathbf{Tag}_c.$$

This implicitly sets $s_2 = c_3$ and $\kappa = \kappa' - c_3$.

If $T = e(g_1, g_2)^{c_1 c_2 c_3}$, this has properly simulated Game_{q_t}. Otherwise, if T is random, a random value is added into M_β. Hence, it has properly simulated Game_{final}.

7 New Schemes

We provide instances for our encoding to achieve new PE schemes. The instances of Inner Product Encryption (IPE) with short keys, Dual Spatial Encryption (Dual SE) with short keys and HIBE with short ciphertexts will be presented. Inner Product Encryption (IPE) with short keys and Dual Spatial Encryption (Dual SE) are new instances. HIBE with short ciphertexts is also found in [32], [12], but applying this instance to our compilers results in new schemes both in asymmetric and symmetric bilinear maps. It should be noted that security analysis of each scheme is replaced by showing that the corresponding instance satisfies the properties that tag based encoding requires.

Inner Product Encryption with short keys

Let define $\mathcal{X} = \mathcal{Y} := \mathbb{Z}_p^\ell$. For all, $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$, $R(x, y) = 1$ iff $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.

- ℓ is the size of a predicate and $\mathbf{h} \in \mathbb{Z}_p^\ell$.
- $\mathbf{kE}(\mathbf{x}, \mathbf{h}) := \langle \mathbf{h}, \mathbf{x} \rangle \in \mathbb{Z}_p$
- $\mathbf{cE}(\mathbf{y}, \mathbf{h}) := (-h_1(y_2/y_1) + h_2, ..., -h_1(y_\ell/y_1) + h_\ell) \in \mathbb{Z}_p^{\ell-1}$
- **Reconstruction:** $\mathbf{m}_x = 1$ and $\mathbf{m}_y = (x_2, ..., x_\ell)$.
- **Linearity:** Firstly, the linearity of $\mathbf{kE}$ holds trivially since $\langle \mathbf{h}, \mathbf{x} \rangle + \langle \mathbf{h}', \mathbf{x} \rangle = \langle \mathbf{h} + \mathbf{h}', \mathbf{x} \rangle$. Also, $\mathbf{cE}(\mathbf{y}, \mathbf{h}) + \mathbf{cE}(\mathbf{y}, \mathbf{h}') = \mathbf{cE}(\mathbf{y}, \mathbf{h} + \mathbf{h}')$ since, for all $i \in [1, \ell - 1]$, $-h_1(y_{i+1}/y_1) + h_{i+1} - h'_1(y_{i+1}/y_1) + h'_{i+1} = -(h_1 + h'_1)(y_{i+1}/y_1) + h_{i+1} + h'_{i+1}$.
- ***h*-hiding:** In the following equation, the first $\ell - 1$ coordinates of the right hand vector in the above equation are independent from the last coordinate by ℓ-*wise independence* [4]. Hence, sharing $\mathbf{h}$ between $\mathbf{kE}, \mathbf{cE}$ is hidden to the adversary.

$$\begin{pmatrix} -y_2/y_1 & 1 & & \\ \vdots & & \ddots & \\ -y_\ell/y_1 & & & 1 \\ x_1 & x_2 & x_3 \cdots & x_\ell \end{pmatrix} \begin{pmatrix} h_1 \\ \vdots \\ h_{\ell-1} \\ h_\ell \end{pmatrix} = \begin{pmatrix} -h_1(y_2/y_1) + h_2 \\ \\ \vdots \\ -h_1(y_\ell/y_1) + h_\ell \\ \langle \mathbf{h}, \mathbf{x} \rangle \end{pmatrix}$$

Dual Spatial Encryption with short keys

For a matrix $M \in \mathbb{Z}_p^{(\ell-1)\times d}$ and a vector $\mathbf{c} \in \mathbb{Z}_p^{\ell-1}$, it defines the affine space $\mathrm{Aff}(M, \mathbf{c}) = \{M\mathbf{w} + \mathbf{c} | \mathbf{w} \in \mathbb{Z}_p^d\}$. Then, $R(\mathbf{x}, \mathrm{Aff}(M, \mathbf{c})) = 1$ iff there exists $\mathbf{w} \in \mathbb{Z}_p^d$ such that $M\mathbf{w} + \mathbf{c} = \mathbf{x}$.

- ℓ is the number of rows of an affine matrix (+1) and $\mathbf{h} = (u_0, \mathbf{u}) \in \mathbb{Z}_p^\ell$.
- $\mathbf{kE}(\mathbf{x}, \mathbf{h}) := u_0 + \mathbf{x}^\intercal \mathbf{u} \in \mathbb{Z}_p$.
- $\mathbf{cE}(\mathrm{Aff}(M, \mathbf{c}), \mathbf{h}) := (u_0 + \mathbf{c}^\intercal \mathbf{u}, M^\intercal \mathbf{u}) \in \mathbb{Z}_p^{d+1}$
- **Reconstruction:** $\mathbf{m}_x = 1$ and $\mathbf{m}_y = (1, \tilde{\mathbf{w}}^\intercal)$ where $\tilde{\mathbf{w}} \in \mathbb{Z}_p^d$ s.t. $M\tilde{\mathbf{w}} + \mathbf{c} = \mathbf{x}$.
- **Linearity:** All coordinates of $\mathbf{kE}(\mathbf{x}, \mathbf{h})$ and $\mathbf{cE}(\mathrm{Aff}(M, \mathbf{c}), \mathbf{h})$ are linear over $\mathbf{h}$.
- ***h*-hiding:** In the following equation, for $\mathbf{x} \in \mathcal{X}$, there is no $\mathbf{w}$ such that $M\mathbf{w} + \mathbf{c} = \mathbf{y}$ since $R(\mathbf{x}, \mathrm{Aff}(M, \mathbf{c})) = 0$. Hence, the last row of the matix on the left is linearly independent from the other rows. Hence, it is hidden that they share u_0 and $\mathbf{u}$.

$$\begin{pmatrix} 1 & \mathbf{c}^\intercal \\ \mathbf{0} & M^\intercal \\ 1 & \mathbf{x}^\intercal \end{pmatrix} \begin{pmatrix} u_0 \\ \mathbf{u} \end{pmatrix} = \begin{pmatrix} u_0 + \mathbf{c}^\intercal \mathbf{u} \\ M^\intercal \mathbf{u} \\ u_0 + \mathbf{x}^\intercal \mathbf{u} \end{pmatrix}$$

HIBE with short ciphertexts [32, 12]

For a vector $\mathbf{ID}_d := (id_1, ..., id_d) \in \mathbb{Z}_p^d$ and a vector $\mathbf{ID}'_{d'} := (id'_1, ..., id'_{d'}) \in \mathbb{Z}_p^{d'}$, $R(\mathbf{ID}_d, \mathbf{ID}'_{d'}) = 1$ iff $d \leq d'$ and $id_i = id'_i \quad \forall i \in [1, d]$.

- ℓ is the maximum depth of an identity (+1) and $\mathbf{h} = (h_0, ..., h_\ell) \in \mathbb{Z}_p^\ell$.
- $\mathbf{kE}(\mathbf{ID}_d, \mathbf{h}) := (h_0 + h_1(id_1) + ... + h_d(id_d), h_{d+1}, ..., h_\ell) \in \mathbb{Z}_p^{\ell-d}$
- $\mathbf{cE}(\mathbf{ID}'_{d'}, \mathbf{h}) := h_0 + h_1(id'_1) + ... + h_{d'}(id'_{d'}) \in \mathbb{Z}_p$
- **Reconstruction:** $\mathbf{m}_x = (1, id'_{d+1}, ..., id'_{d'}, 0, ..., 0) \ \in \mathbb{Z}_p^{\ell-d}$ and $\mathbf{m}_y = 1$.
- **Linearity:** All coordinates of $\mathbf{kE}(\mathbf{ID}_d, \mathbf{h})$ and $\mathbf{cE}(\mathbf{ID}'_d, \mathbf{h})$ are linear over $\mathbf{h}$.
- ***h*-hiding:** In the following equation, the first $\ell - d + 1$ rows are linearly independent with the last row of matrix on the left since $id_d \neq id'_d$ and $h_0, ..., h_\ell$ appear at most twice. Therefore, the sharing $\mathbf{h}$ between the first $\ell + 1$ coordinates of the vector of the right hand of the equation with the last coordinate of the vector is hidden.

$$\begin{pmatrix} 1 & id_1 & \cdots & id_d & & & \\ & & & 1 & & & \\ & & & & & \ddots & \\ & & & & & & 1 \\ 1 & id'_1 & \cdots & id'_d & \cdots & id'_{d'} & \end{pmatrix} \begin{pmatrix} h_0 \\ \vdots \\ h_{\ell-1} \\ h_\ell \end{pmatrix} = \begin{pmatrix} h_0 + h_1(id_1) + ... + h_d(id_d) \\ h_{d+1} \\ \vdots \\ h_\ell \\ h_0 + h_1(id'_1) + ... + h_{d'}(id'_{d'}) \end{pmatrix}$$

8 Conclusion

In this paper, we proposed a new encoding framework for PE schemes. Our framework provides an encryption scheme having a better efficiency when the size of their encoding is large compared with prior encoding frameworks. We provided two generic constructions for our framework as compilers of encodings. They are adaptively secure under the standard assumption. Consequently, we showed that our encoding is versatile by proposing a number of new instances that are applicable to our encodings.

References

1. S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In E. Kushilevitz and T. Malkin, editors, *TCC*, volume 9563 of *LNCS*, pages 259–288. Springer, 2016.
2. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 557–577. Springer, 2014.
3. N. Attrapadung. Dual system encryption framework in prime-order groups. *IACR Cryptology ePrint Archive*, 2015:390, 2015.
4. N. Attrapadung and B. Libert. Functional encryption for public-attribute inner products: Achieving constant-size ciphertexts with adaptive security or support for negation. *J. Mathematical Cryptology*, 5(2):115–158, 2012.
5. N. Attrapadung and S. Yamada. Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In K. Nyberg, editor, *CT-RSA*, volume 9048 of *LNCS*, pages 87–105. Springer, 2015.

6. A. Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
7. X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In C. Dwork, editor, *CRYPTO*, volume 4117 of *LNCS*, pages 290–307. Springer, 2006.
8. C. Chen, Z. Zhang, and D. Feng. Fully secure doubly-spatial encryption under simple assumptions. In T. Takagi, G. Wang, Z. Qin, S. Jiang, and Y. Yu, editors, *ProvSec*, volume 7496 of *LNCS*, pages 253–263. Springer, 2012.
9. J. Chen, R. Gay, and H. Wee. Improved dual system ABE in prime-order groups via predicate encodings. In E. Oswald and M. Fischlin, editors, *EUROCRYPT*, volume 9057 of *LNCS*, pages 595–624. Springer, 2015.
10. J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In R. Canetti and J. A. Garay, editors, *CRYPTO*, volume 8043 of *LNCS*, pages 435–460. Springer, 2013.
11. J. Chen and H. Wee. Doubly spatial encryption from DBDH. *Theor. Comput. Sci.*, 543:79–89, 2014.
12. J. Chen and H. Wee. Dual system groups and its applications - compact HIBE and more. *IACR Cryptology ePrint Archive*, 2014:265, 2014.
13. P. Datta, R. Dutta, and S. Mukhopadhyay. Fully secure self-updatable encryption in prime order bilinear groups. In S. S. M. Chow, J. Camenisch, L. C. K. Hui, and S. Yiu, editors, *ISC*, volume 8783 of *LNCS*, pages 1–18. Springer, 2014.
14. A. Escala, G. Herold, E. Kiltz, C. Ràfols, and J. L. Villar. An algebraic framework for diffie-hellman assumptions. In R. Canetti and J. A. Garay, editors, *CRYPTO*, volume 8043 of *LNCS*, pages 129–147. Springer, 2013.
15. D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *LNCS*, pages 44–61. Springer, 2010.
16. A. Guillevic. Comparing the pairing efficiency over composite-order and prime-order elliptic curves. In M. J. J. Jr., M. E. Locasto, P. Mohassel, and R. Safavi-Naini, editors, *ACNS*, volume 7954 of *LNCS*, pages 357–372. Springer, 2013.
17. M. Hamburg. Spatial encryption. *IACR Cryptology ePrint Archive*, 2011:389, 2011.
18. G. Herold, J. Hesse, D. Hofheinz, C. Ràfols, and A. Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In *CRYPTO*.
19. K. Lee, S. G. Choi, D. H. Lee, J. H. Park, and M. Yung. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency. In *ASIACRYPT*.
20. A. B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In D. Pointcheval and T. Johansson, editors, *EUROCRYPT*, volume 7237 of *LNCS*, pages 318–335. Springer, 2012.
21. A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In H. Gilbert, editor, *EUROCRYPT*, volume 6110 of *LNCS*, pages 62–91. Springer, 2010.
22. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. *IACR Cryptology ePrint Archive*, 2009:482, 2009.
23. A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *LNCS*, pages 180–198. Springer, 2012.
24. T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 214–231. Springer, 2009.
25. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO*, volume 6223 of *LNCS*, pages 191–208. Springer, 2010.

26. T. Okamoto and K. Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. In D. Lin, G. Tsudik, and X. Wang, editors, *CANS*, volume 7092 of *LNCS*, pages 138–159. Springer, 2011.
27. T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In X. Wang and K. Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 349–366. Springer, 2012.
28. S. C. Ramanna. More efficient constructions for inner-product encryption. In M. Manulis, A. Sadeghi, and S. Schneider, editors, *ACNS*, volume 9696 of *LNCS*, pages 231–248. Springer, 2016.
29. S. C. Ramanna, S. Chatterjee, and P. Sarkar. Variants of waters' dual system primitives using asymmetric pairings - (extended abstract). In M. Fischlin, J. A. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *LNCS*, pages 298–315. Springer, 2012.
30. J. H. Seo. On the (im)possibility of projecting property in prime-order setting. In X. Wang and K. Sako, editors, *ASIACRYPT*, volume 7658 of *LNCS*, pages 61–79. Springer, 2012.
31. B. Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In S. Halevi, editor, *CRYPTO*, volume 5677 of *LNCS*, pages 619–636. Springer, 2009.
32. H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC*, volume 8349 of *LNCS*, pages 616–637. Springer, 2014.

A Our Compiler with Symmetric Bilinear maps

A.1 Complexity Assumptions under Symmetric Bilinear maps

The adaptive security of our framework relies on both *DBDH* and *DLIN* whose definitions are reviewed below.

Decisional Bilinear Diffie-Hellman (*DBDH*) assumption. Let g be a generator of G and c_1, c_2, c_3 be selected randomly from $\mathbb{Z}_p$. Given $\{g, g^{c_1}, g^{c_2}, g^{c_3} \in G, T \in G_T\}$, there is no PPT algorithm that can distinguish whether T is $e(g,g)^{c_1 c_2 c_3}$ or a random from G_T with a non-negligible advantage.

Decisional Linear (*DLIN*) assumption Let g, f, ν be random generators of G and c_1, c_2 be selected randomly from $\mathbb{Z}_p$. Given $\{g, f, \nu, g^{c_1}, f^{c_2}, T \in G\}$, there is no PPT algorithm can distinguish whether T is $\nu^{c_1+c_2}$ or a random from G with a non-negligible advantage.

We remark that the *DBDH* assumption can be reduced to the *DLIN* assumption [11, 7]. This implies that our construction is secure under the *DLIN* assumption.

A.2 The Construction

For a tag based encoding **TE** for a predicate R where $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$, with ℓ which is an integer to associated with R, **PE(TE)** is constructed by four algorithms following:

- Setup(1^λ,ℓ): The algorithm randomly generates two groups G and G_T from $\mathcal{G}(\lambda, p)$. Next, it generates $g, v, v_1, v_2, w \in G$ and exponents α, a_1, a_2, b, h_1, …, $h_\ell \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$ and $\mathbf{h} = (h_1, ..., h_\ell)$. It publishes the public parameters PK as follows

$$(g, g^b, g^{a_1}, g^{a_2}, g^{b \cdot a_1}, g^{b \cdot a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, g^{\mathbf{h}}, e(g,g)^{\alpha \cdot a_1 \cdot b})$$

  The MSK consists of $(g^\alpha, g^{\alpha \cdot a_1}, v, v_1, v_2)$.
- Keygen(MSK, PK, x): The algorithm chooses randomly $r_1, r_2, z_1, z_2, h'_1, ..., h'_\ell \in \mathbb{Z}_p$ and sets $r = r_1 + r_2$ and $\mathbf{Tag}_k = \mathbf{kE}(x, \mathbf{h}')$ where $\mathbf{h}'$ is equal to $(h'_1, ..., h'_\ell)$. Then, it creates SK as follows

$$D_1 = g^{\alpha \cdot a_1} v^r, D_2 = g^{-\alpha} v_1^r g^{z_1}, D_3 = (g^b)^{-z_1}, D_4 = v_2^r g^{z_2}, D_5 = (g^b)^{-z_2},$$

$$D_6 = g^{r_2 \cdot b}, D_7 = g^{r_1}, \mathbf{K} = (g^{\mathbf{kE}(x,\mathbf{h})} w^{\mathbf{Tag}_k})^{r_1}.$$

  It sets $SK = (D_1, ..., D_7, \mathbf{K}, \mathbf{Tag}_k)$.
- Encrypt(PK, M, y): The algorithm selects s_1, s_2, t, $h''_1, ..., h''_\ell \in \mathbb{Z}_p$ and set $s = s_1 + s_2$, and $\mathbf{Tag}_c = \mathbf{cE}(y, \mathbf{h}'')$ where $\mathbf{h}''$ is equal to $(h''_1, ..., h''_\ell)$. It creates the ciphertext, CT as follows

$$C = M \cdot (e(g,g)^{\alpha a_1 \cdot b})^{s_2}, C_1 = (g^b)^s, C_2 = (g^{b \cdot a_1})^{s_1}, C_3 = (g^{a_1})^{s_1}, C_4 = (g^{b \cdot a_2})^{s_2},$$

$$C_5 = (g^{a_2})^{s_2}, C_6 = \tau_1^{s_1} \tau_2^{s_2}, C_7 = (\tau_1^b)^{s_1} (\tau_2^b)^{s_2} w^{-t}, C_8 = g^t, \mathbf{E} = (g^{\mathbf{cE}(y,\mathbf{h})} w^{\mathbf{Tag}_c})^t$$

It sets $CT = (C, C_1, ..., C_8, \mathbf{E}, \mathbf{Tag}_c)$.

- Decrypt(x, y, SK, CT, PK): First, the algorithm calculates

$$A_1 = e(C_1, D_1)e(C_2, D_2)e(C_3, D_3)e(C_4, D_4)e(C_5, D_5),\ A_2 = e(C_6, D_6)e(C_7, D_7).$$

Since $R(x, y) = 1$, there exist reconstruction vectors $\mathbf{m}_x$ and $\mathbf{m}_y$ such that $\mathbf{m}_x \mathbf{kE}(x, \mathbf{h}) = \mathbf{m}_y \mathbf{cE}(y, \mathbf{h})$. If $\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c$ is 0, it aborts. Otherwise,

$$A_3 = e(C_8, \mathbf{K}^{\mathbf{m}_x})/e(\mathbf{E}^{\mathbf{m}_y}, D_7) = e(g, w)^{r_1 t \cdot (\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)}.$$

Therefore, $M = C \cdot A_2/(A_1 \cdot A_3^{1/(\mathbf{m}_x \mathbf{Tag}_k - \mathbf{m}_y \mathbf{Tag}_c)})$.

Correctness of the construction is almost identical to that of the construction with asymmetric pairing in the main contents. The number of pairing computations of this construction also can be reduced as the construction with asymmetric pairing does.

Theorem 2. *Suppose there exists Tag based Encoding* **TE***, then our* **PE(TE)** *is adaptively secure under* DLIN *assumption.*

Proof. This is proved by lemmas 4, 5 and 6.

A.3 Security Analysis

Semi-functional Ciphertext. By running Encrypt algorithm for a message M and an input y, the algorithm generates a normal ciphertext $CT = (C', C_1', ..., C_8', \mathbf{E}', \mathbf{Tag}_c')$. Then, it randomly selects $\kappa \in \mathbb{Z}_p$ and sets

$$C = C', \quad C_1 = C_1', \quad C_2 = C_2', \quad C_3 = C_3', \quad C_4 = C_4' g^{b a_2 \kappa},$$

$$C_5 = C_5' g^{a_2 \kappa},\ C_6 = C_6' v_2^{a_2 \kappa},\ C_7 = C_7' v_2^{a_2 b \kappa},\ C_8 = C_8',\ \mathbf{E} = \mathbf{E}',\ \mathbf{Tag}_c = \mathbf{Tag}_c'.$$

Semi-functional Key. The algorithm generates a normal key $SK = (D_1', \ldots, D_7', \mathbf{K}', \mathbf{Tag}_k')$ by running Keygen algorithm for an input x. Then, it sets

$$D_1 = D_1' g^{-a_1 a_2 \gamma}, \quad D_2 = D_2' g^{a_2 \gamma}, \quad D_3 = D_3', \quad D_4 = D_4' g^{a_1 \gamma}$$

$$D_5 = D_5', \quad D_6 = D_6', \quad D_7 = D_7', \quad \mathbf{K} = \mathbf{K}', \quad \mathbf{Tag}_k = \mathbf{Tag}_k'.$$

It should be noted that $e(g, g)^{a_1 a_2 b \kappa \gamma}$ will be added to the message to be encrypted if the semi-functional key is used for decrypting the semi-functional ciphertext.

Security Games The definitions of security games and the strategy to prove adaptive security are identical to those of the compiler in asymmetric pairings.

Lemma 4. (Semi-functional Ciphertext Invariance) *Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes* Game_{real} *and* Game_0 *with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* DLIN *assumption with advantage ϵ.*

Proof. Firstly, $\mathcal{B}$ takes an $\{g, f, \nu, g^{c_1}, f^{c_2}, T\}$ as an instance from *DLIN* assumption. Depending on the value of T, $\mathcal{B}$ will simulate Game_{real} or Game_0 to take an advantage from $\mathcal{A}$ which can distinguish both games with non-negligible advantage ϵ.

Setup Algorithm $\mathcal{B}$ chooses exponents $\alpha, b, y_v, y_{v_1}, y_{v_2}, y_w, h_1, ...h_\ell$ randomly from $\mathbb{Z}_p$ and group elements w, g. Then, it publishes public parameters as:

$$\text{PK} := \{g,\ g^b,\ g^{a_1} = f,\ g^{a_2} = \nu,\ g^{b\cdot a_1} = f^b,\ g^{b\cdot a_2} = \nu^b,\ w = g^{y_w}, g^{h_1}, ..., g^{h_\ell},$$

$$\tau_1 = g^{y_v} f^{y_{v_1}},\ \tau_2 = g^{y_v} \nu^{y_{v_2}},\ \tau_1^b,\ \tau_2^b,\ e(g,g)^{\alpha b a_1} = e(g,f)^{\alpha b}\}$$

it implies that $v = g^{y_v}$, $v_1 = g^{y_{v_1}}$, $v_2 = g^{y_{v_2}}$. It sets $\text{MSK} := \{g^\alpha, g^{\alpha\cdot a_1} = f^\alpha\}$.

Phase I and II Because $\mathcal{B}$ knows the public parameters and the master secret key, it can generate normal keys using Keygen.

Challenge Ciphertext When the adversary requests the challenge ciphertext for y^* with messages M_0, M_1, $\mathcal{B}$ randomly selects β from $\{0,1\}$. $\mathcal{B}$ runs the encryption algorithm to generate a normal ciphertext $CT' = (C', C_1', ..., C_8', \mathbf{E}', \mathbf{Tag}_c')$ for y^* and M_β. We denote the random exponents of CT' as s_1', s_2', t'. Then it sets the challenge CT as:

$$C = C'\cdot(e(g^{c_1}, f)\cdot e(g, f^{c_2}))^{b\cdot\alpha},\ C_1 = C_1'\cdot(g^{c_1})^b,\ C_2 = C_2'\cdot(f^{c_2})^{-b},\ C_3 = C_3' f^{c_2},$$

$$C_4 = C_4'(T)^b, \quad C_5 = C_5'\cdot T, \quad C_6 = C_6'\cdot(g^{c_1})^{y_v}\cdot(f^{c_2})^{-y_{v_1}}\cdot T^{y_{v_2}},$$

$$C_7 = C_7'\cdot((g^{c_1})^{y_v}\cdot(f^{c_2})^{-y_{v_1}}\cdot T^{y_{v_2}})^b, \quad C_8 = C_8', \quad \mathbf{E} = \mathbf{E}', \quad \mathbf{Tag}_c = \mathbf{Tag}_c'.$$

This implicitly sets $s_1 = -c_2 + s_1'$, $s_2 = s_2' + c_1 + c_2$ and $s = s_1 + s_2 = c_1 + s_1' + s_2'$. If $T = \nu^{c_1+c_2}$, the challenge ciphertext is a normal ciphertext, and $Game_{real}$ has been simulated properly. Otherwise, if T is a random, and is denoted as $\nu^{c_1+c_2}\nu^\kappa$, this is a properly distributed semi-functional key, and $Game_0$ has been simulated properly.

Lemma 5. (Semi-functional Key Invariance) *Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes Game_{k-1} and Game_k with a non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* DLIN *assumption with ϵ.*

Proof. Firstly, $\mathcal{B}$ takes an $\{g, f, \nu, g^{c_1}, f^{c_2}, T\}$ as an instance from *DLIN* assumption. Depending on the value of T, $\mathcal{B}$ will simulate Game_{k-1} or Game_k to take an advantage from $\mathcal{A}$ which can distinguish between both games with non-negligible advantage.

Setup Algorithm $\mathcal{B}$ chooses exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, h_1', ...h_\ell', \tilde{h}_1, ...\tilde{h}_\ell$, randomly from $\mathbb{Z}_p$. Then, it sets

$$g = g, g^b = f, g^{b\cdot a_1} = f^{a_1}, g^{b\cdot a_2} = f^{a_2}, v = \nu^{-a_1 a_2}, v_1 = \nu^{a_2}\cdot g^{y_{v_1}}, v_2 = \nu^{a_1}\cdot g^{y_{v_2}},$$

$$\tau_1 = v v_1^{a_1} = g^{y_{v_1} a_1}, \tau_2 = v v_1^{a_2} = g^{y_{v_2} a_2}, \tau_1^b = f^{y_{v_1} a_1}, \tau_2^b = f^{y_{v_2} a_2},$$

$$w = f g^{y_w}, g^{h_i} = f^{-h_i'} g^{\tilde{h}_i} \forall i \in [1, \ell],\ e(g,g)^{\alpha\cdot a_1 b} = e(f,g)^{\alpha\cdot a_1}.$$

It publishes the public parameters following

$$(g, g^b, g^{a_1}, g^{a_2}, g^{b\cdot a_1}, g^{b\cdot a_2}, \tau_1, \tau_2, \tau_1^b, \tau_2^b, w, g^{\mathbf{h}}, e(g,g)^{\alpha\cdot a_1\cdot b})$$

where $\mathbf{h} = (h_1, ..., h_\ell)$. In the setting, $\mathbf{h}$ is implicitly set by $\tilde{\mathbf{h}} - y_f\mathbf{h}'$ where $\mathbf{h}' = (h'_1, ...h'_\ell)$ and $\tilde{\mathbf{h}} = (\tilde{h}_1, ...\tilde{h}_\ell)$ if we write $f = g^{y_f}$. Because $\mathcal{B}$ does not know y_f, it cannot calculate the values of $\mathbf{h}$, but it calculates $g^{\mathbf{h}}$ because it knows $g, f, \tilde{\mathbf{h}}, \mathbf{h}'$. It should be noted that, initially, the values of $\{h'_i; \forall i \in [1, \ell]\}$ are not revealed, which means that they are initially information theoretically hidden because, for all i, $\tilde{h}_i$ is uniquely added where h'_i appears. $\mathcal{B}$ knows all MSK $= (g^\alpha, g^{\alpha \cdot a_1}, v, v_1, v_2)$ since it knows α and a_1.

Phase I and II For the first $k-1$ semi-functional keys, $\mathcal{B}$ generates a normal key and selects γ randomly from $\mathbb{Z}_p$. It then adds semi-functional parts to the normal key . This is possible because $\mathcal{B}$ knows all public parameters and MSK. Similarly, for the rest keys except k^{th} key $(i > k)$, $\mathcal{B}$ can generate normal keys using the key generation algorithm, **KeyGen**, as the same reasons.

For the k^{th} key, $\mathcal{B}$ sets $\mathbf{Tag}_k = \mathbf{kE}(x, \mathbf{h}')$. Then, with $\mathbf{Tag}'_k$, it generates a normal key $SK' = (D'_1, ..., D'_7, \mathbf{K}', \mathbf{Tag}_k)$ using the key generation algorithm. Then, it sets the k^{th} key as follows

$$D_1 = D'_1 T^{-a_1 a_2},\ D_2 = D'_2 T^{a_2} (g^{c_1})^{y_{v_1}},\ D_3 = D'_3 (f^{c_2})^{y_{v_1}},\ D_4 = D'_4 T^{a_1} (g^{c_1})^{y_{v_2}},$$

$$D_5 = D'_5 (f^{c_2})^{y_{v_2}}, D_6 = D'_6 f^{c_2}, D_7 = D'_7 (g^{c_1}), \mathbf{K} = \mathbf{K}' (g^{c_1})^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')}, \mathbf{Tag}_k.$$

We let r'_1, r'_2, z'_1, z'_2 denote the random exponents of SK'. Then, it implicitly sets $z_1 = z'_1 - y_{v_1} c_2$ and $z_2 = z'_2 - y_{v_2} c_2$. Also, by *linearity* property,

$$g^{\mathbf{kE}(x, \mathbf{h})} = g^{\mathbf{kE}(x, -y_f \mathbf{h}' + \tilde{\mathbf{h}})} = g^{\mathbf{kE}(x, -y_f \mathbf{h}')} g^{\mathbf{kE}(x, \tilde{\mathbf{h}})} = f^{-\mathbf{kE}(x, \mathbf{h}')} g^{\mathbf{kE}(x, \tilde{\mathbf{h}})}$$

Therefore, the value of $\mathbf{K}'$ can be represented as follows:

$$\mathbf{K}' = (f^{-\mathbf{kE}(x, \mathbf{h}')} g^{\mathbf{kE}(x, \tilde{\mathbf{h}})} (f g^{y_w})^{\mathbf{kE}(x, \mathbf{h}')})^{r'_1} = (g^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{r'_1}.$$

This implies that $\mathbf{K} = \mathbf{K}' (g^{c_1})^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')} = (g^{\mathbf{kE}(x, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{r'_1 + c_1}$.

If T is equal to $\nu^{c_1 + c_2}$, then the k^{th} key is a normal key with $r_1 = r'_1 + c_1$ and $r_2 = r'_2 + c_2$. Otherwise, if T is $\nu^{c_1 + c_2} g^\gamma$, which means a random group element, then, the k^{th} key is a properly distributed semi-functional key.

Challenge Ciphertext When the adversary requests the challenge ciphertext for y^* with messages M_0, M_1, $\mathcal{B}$ randomly selects β from $\{0, 1\}$. With $\mathbf{Tag}_c = \mathbf{cE}(y^*, \mathbf{h}')$, $\mathcal{B}$ runs the encryption algorithm to generate a normal ciphertext $CT' = (C', C'_1, ..., C'_8, \mathbf{E}', \mathbf{Tag}_c)$ for y^* and M_β. We let s'_1, s'_2, t' denote the random exponents of CT'. To make the semi-functional challenge ciphertext, it randomly selects $\kappa \in \mathbb{Z}_p$ and sets $C = C', C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$. Additionally, it sets

$$C_4 = C'_4 f^{a_2 \cdot \kappa}, \quad C_5 = C'_5 \cdot g^{a_2 \cdot \kappa}, \quad C_6 = C'_6 \cdot v_2^{a_2 \kappa}, C_7 = C'_7 \cdot f^{y_{v_2} \cdot \kappa \cdot a_2} \nu^{-a_1 \cdot \kappa \cdot y_w \cdot a_2}$$

$$C_8 = g^{t'} \cdot \nu^{a_1 a_2 \kappa}, \quad \mathbf{E} = \mathbf{E}' \cdot (\nu^{\mathbf{cE}(y^*, \tilde{\mathbf{h}} + y_w \mathbf{h}')})^{a_1 a_2 \kappa}, \quad \mathbf{Tag}_c$$

This implicitly sets $g^t = g^{t'} \cdot \nu^{a_1 a_2 \kappa}$. It should be noted that $\nu^{a_1 a_2 b \kappa}$ of $v_2^{a_2 b \kappa}$ is cancelled out by w^{-t} in C_7. The fact that $\mathbf{Tag}_c$ in the challenge ciphertext and $\mathbf{Tag}_k$ in the k^{th} key share the same vector $\mathbf{h}'$ is hidden to the adversary by *h-hiding* property since $R(x, y^*) = 0$. Also, $\mathbf{E}$ is a valid ciphertext elements since

$$\mathbf{E}' = (f^{-\mathbf{cE}(y^*,\mathbf{h}')} g^{\mathbf{cE}(y^*,\tilde{\mathbf{h}})} (f g^{y_w})^{\mathbf{cE}(y^*,\mathbf{h}')})^{t'} = (g^{\mathbf{cE}(y^*,\tilde{\mathbf{h}}+y_w\mathbf{h}')})^{t'}.$$

The second equality of the above equation holds by *linearity* property.

In the simulation, $\mathcal{B}$ cannot test whether the k^{th} key is normal or semi-functional by generating a ciphertext which can be decrypted only by a normal key because $\mathbf{Tag}_k$ and $\mathbf{Tag}_c$ must share $\mathbf{h}'$. In our simulation, $\mathbf{m}_x\mathbf{Tag}_k - \mathbf{m}_{y^*}\mathbf{Tag}_c = 0$ if the simulator generates a valid semi-functional ciphertext such that $R(x,y) = 1$. Hence, the decryption algorithm will abort.

Lemma 6. (Semi-functional Security) *Suppose that there exists an algorithm $\mathcal{A}$ which distinguishes Game_{q_t} and Game_{final} with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks* DBDH *assumption with advantage ϵ.*

Proof. First, $\mathcal{B}$ takes an $\{g, g^{c_1}, g^{c_2}, g^{c_3}, T\}$ as an instance from *DLIN* assumption. Depending on the value of T, $\mathcal{B}$ will simulate Game_{q_t} or Game_{final} to take an advantage from $\mathcal{A}$ which can distinguish both games with non-negligible advantage ϵ.

Setup $\mathcal{B}$ chooses exponents $b, a_1, y_v, y_{v_1}, y_{v_2}, y_w, h_1, ...h_\ell$, randomly from $\mathbb{Z}_p$, and sets

$$g = g,\ g^b,\ g^{a_1}, g^{a_2} = g^{c_2},\ g^{b\cdot a_1},\ g^{b\cdot a_2} = (g^{c_2})^b, w = g^{y_w}, \{g^{h_i};\ \forall i \in [1,\ell]\}$$

$$\tau_1 = g^{y_v + y_{v_1} a_1}, \tau_2 = g^{y_v}(g^{c_2})^{y_{v_2}}, \tau_1^b, \tau_2^b, e(g,g)^{\alpha\cdot a_1 b} = e(g^{c_1}, g^{c_2})^{b a_1}.$$

It implicitly sets $v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}, \alpha = c_1 \cdot c_2, a_2 = c_2$. It should be noted that MSK cannot be explicitly calculated because $\mathcal{B}$ does not know g^α.

Phase I and II For semi-functional keys, if the adversary requests a private key for x, it randomly generates r_1, r_2, z_1, z_2, γ', $h'_1, ..., h'_\ell$ from $\mathbb{Z}_p$ and sets $\mathbf{Tag}_k = \mathbf{kE}(x, \mathbf{h}')$ where $\mathbf{h}' = (h'_1, .., h'_\ell)$. It, then, creates semi-functional key as follows:

$$D_1 = (g^{c_2})^{-\gamma' a_1} v^r,\ D_2 = (g^{c_2})^{\gamma'} v_1^r g^{z_1},\ D_3 = (g^b)^{-z_1},\ D_4 = (g^{c_1})^{a_1} g^{a_1\cdot\gamma'} v_2^r g^{z_2},$$

$$D_5 = (g^b)^{-z_2}, \quad D_6 = g^{r_2\cdot b}, \quad D_7 = g^{r_1}, \quad \mathbf{K} = (g^{\mathbf{kE}(x,\mathbf{h})} w^{\mathbf{Tag}_k})^{r_1}, \quad \mathbf{Tag}_k.$$

where $\mathbf{h} = (h_1, .., h_\ell)$. This implicitly sets $\gamma = c_1 + \gamma'$ and $r = r_1 + r_2$.

Challenge Ciphertext When the adversary requests the challenge ciphertext for y^* with messages M_0, M_1, $\mathcal{B}$ randomly selects β from $\{0,1\}$. Then it generates random values $\kappa', s_1, t, h''_1, ...h''_\ell$ from $\mathbb{Z}_p$ and sets $\mathbf{Tag}_c = \mathbf{cE}(y^*, \mathbf{h}'')$ where $\mathbf{h}'' = (h''_1, ...h''_\ell)$. It, then, creates the challenge ciphertext as follows:

$$C = M_\beta T^{a_1 b}, C_1 = g^{s_1\cdot b} + (g^{c_3})^b, C_2 = g^{b\cdot a_1\cdot s_1}, C_3 = g^{a_1\cdot s_1}, C_4 = (g^{c_2})^{\kappa'\cdot b},$$

$$C_5 = (g^{c_2})^{\kappa'}, C_6 = \tau_1^{s_1}(g^{c_3})^{y_v}(g^{c_2})^{y_{v_2}\cdot\kappa'},\ C_7 = (\tau_1^b)^{s_1}(g^{c_3})^{y_v\cdot b}(g^{c_2})^{y_{v_2}\cdot\kappa'\cdot b} w^{-t},$$

$$C_8 = g^t,\ \mathbf{E} = (g^{\mathbf{cE}(y^*,\mathbf{h}'')} w^{\mathbf{Tag}_c}), \mathbf{Tag}_c.$$

This implicitly sets $s_2 = c_3$ and $\kappa = \kappa' - c_3$.

In this simulation, if $T = e(g,g)^{c_1 c_2 c_3}$, then this has properly simulated Game_{q_t}. Otherwise, if T is random, a random value is added into M_β. Hence, it has properly simulated Game_{final}.

B Instances from Existing Schemes

In this section, we extracts instances from literature. New instances that result Identity based encryption (IBE) [31], Public Attribute Inner Product Encryption (PAIPE) with short ciphertexts [4], Spatial Encryption (SE) [8] and Doubly Spatial Encryption (DSE) [32] are derived from previous works. It should be noted that the schemes created by applying those instances to our compiler with symmetric bilinear maps are identical to the original constructions of their literature.

Waters' IBE is a good example to describe our framework due to its simplicity.

Identity Based Encryption [31]

Let $\mathcal{X} = \mathcal{Y} := \mathbb{Z}_p$. For all $ID \in \mathcal{X}$ and $ID' \in \mathcal{Y}$, $R(ID, ID') = 1$ iff $ID = ID'$.

- ℓ=2 and $\mathbf{h} = (y_u, y_h) \in \mathbb{Z}_p^2$
- $\mathbf{kE}(ID, (y_u, y_h)) := (y_u ID + y_h) \in \mathbb{Z}_p$
- $\mathbf{cE}(ID', (y_u, y_h)) := (y_u ID' + y_h) \in \mathbb{Z}_p$
- **Reconstruction:** This is an exact cancellation. Therefore, $\mathbf{m}_x = \mathbf{m}_y = 1$.
- **Linearity:** For all $\mathbf{h}' = (y'_u, y'_h)$,

$$\begin{aligned}\mathbf{kE}(ID, (y'_u, y'_h)) + \mathbf{kE}(ID, (\tilde{y}_u, \tilde{y}_h)) =& y'_u ID + y'_h + \tilde{y}_u ID + \tilde{y}_h \\ =& \mathbf{kE}(ID, (y'_u + \tilde{y}_u, y'_h + \tilde{y}_h)).\end{aligned}$$

  The linearity of $\mathbf{cE}(ID', (y'_u, y'_h))$ is identical showed with $\mathbf{kE}(ID, (y'_u, y'_h))$.
- ***h*-hiding:** $y_u ID + y_h$ and $y_u ID' + y_h$ are pair-wise independent since $R(ID, ID') = 0$ (i.e. $ID \neq ID'$). Hence, for given $y_u ID + y_h$, the value of $y_u ID' + y_h$ is uniformly distributed from $\mathbb{Z}_p$. Therefore, it is not distinguishable from $y'_u ID' + y'_h$ where (y'_u, y'_v) is randomly selected from $\mathbb{Z}_p^2$.

Public Attribute Inner Product Encryption with short ciphertexts and Spatial Encryption with short ciphertexts are the symmetric conversion of IPE with short keys and Dual SE which short keys which were introduced in the previous section. If we let $\mathbf{kE}'$ and $\mathbf{cE}'$ denote the encodings of the original schemes (e.g. IPE with short keys), then we define $\mathbf{kE} = \mathbf{cE}'$ and $\mathbf{cE} = \mathbf{kE}'$. All properties are identically proved with the schemes in previous sections.

Public Attribute Inner Product Encryption with short ciphertexts [4]

Let $\mathcal{X} = \mathcal{Y} := \mathbb{Z}_p^\ell$. For all, $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$, $R(x, y) = 1$ iff $\langle \mathbf{x}, \mathbf{y} \rangle = 0$

- ℓ is the size of a predicate and $\mathbf{h} \in \mathbb{Z}_p^\ell$
- $\mathbf{kE}(\mathbf{x}, \mathbf{h}) := (\{-h_1(x_i/x_1) + h_i\}_{i=2,\ldots\ell}) \in \mathbb{Z}_p^{\ell-1}$
- $\mathbf{cE}(\mathbf{y}, \mathbf{h}) := \langle \mathbf{h}, \mathbf{y} \rangle \in \mathbb{Z}_p$
- All properties are proved similarly with our IPE with short keys.

Spatial Encryption with short ciphertexts [8]

For a matrix $M \in \mathbb{Z}_p^{(\ell-1)\times d}$ and a vector $\mathbf{c} \in \mathbb{Z}_p^{\ell-1}$, it defines the affine space $\mathrm{Aff}(M, \mathbf{c}) = \{M\mathbf{w} + \mathbf{c} | \mathbf{w} \in \mathbb{Z}_p^d\}$. Then, $R(\mathrm{Aff}(M, \mathbf{c}), \mathbf{y}) = 1$ iff there exists $\mathbf{w} \in \mathbb{Z}_p^d$ such that $M\mathbf{w} + \mathbf{c} = \mathbf{y}$.

- ℓ is the number of rows of an affine matrix (+1) and $\mathbf{h} = (u_0, \mathbf{u}) \in \mathbb{Z}_p^\ell$.
- $\mathbf{kE}(\text{Aff}(M, \mathbf{c}), \mathbf{h}) := (u_0 + \mathbf{c}^\intercal \mathbf{u}, M^\intercal \mathbf{u}) \in \mathbb{Z}_p^{d+1}$
- $\mathbf{cE}(\mathbf{y}, \mathbf{h}) := (u_0 + \mathbf{y}^\intercal \mathbf{u}) \in \mathbb{Z}_p$
- All properties are proved similarly with our Dual SE with short keys.

Doubly Spatial Encryption [8]

For affine matrices $X \in \text{AffM}(\mathbb{Z}_p^{\ell \times d})$ and $Y \in \text{AffM}(\mathbb{Z}_p^{\ell \times f})$, $R(X, Y) = 1$ iff there exist $\mathbf{w} \in \text{AffM}(\mathbb{Z}_p^d)$ and $\mathbf{z} \in \text{AffM}(\mathbb{Z}_p^f)$ such that $\mathbf{w}X^\intercal = \mathbf{z}Y^\intercal$

- ℓ is the number of rows of affine matrices, X and Y and $\mathbf{h} \in \mathbb{Z}_p^\ell$.
- $\mathbf{kE}(X, \mathbf{h}) := (X^\intercal \mathbf{h}^\intercal) \in \mathbb{Z}_p^d$.
- $\mathbf{cE}(Y, \mathbf{h}) := (Y^\intercal \mathbf{h}^\intercal) \in \mathbb{Z}_p^f$.
- **Reconstruction:**
  Since $\mathbf{w}X^\intercal = \mathbf{z}Y^\intercal$, $\mathbf{w} \cdot \mathbf{kE}(X, \mathbf{h}) = \mathbf{z} \cdot \mathbf{cE}(Y, \mathbf{h})$ (i.e. $\mathbf{m}_x = \mathbf{w},$ and $\mathbf{m}_y = \mathbf{z}$).
- **Linearity:** For all $\mathbf{h}$ and $\mathbf{h}'$ from $\mathbb{Z}_p^\ell$,

$$\mathbf{kE}(X, \mathbf{h}) + \mathbf{kE}(X, \mathbf{h}') = X^\intercal \mathbf{h}^\intercal + X\mathbf{h}'^\intercal = X^\intercal(\mathbf{h}^\intercal + \mathbf{h}'^\intercal) = \mathbf{kE}(X, \mathbf{h}^\intercal + \mathbf{h}'^\intercal)$$

$$\mathbf{cE}(Y, \mathbf{h}) + \mathbf{cE}(Y, \mathbf{h}') = Y^\intercal \mathbf{h}^\intercal + Y\mathbf{h}'^\intercal = Y^\intercal(\mathbf{h}^\intercal + \mathbf{h}'^\intercal) = \mathbf{cE}(Y, \mathbf{h}^\intercal + \mathbf{h}'^\intercal).$$

- ***h*-hiding:** In the following equation, there exist no $\mathbf{w}$ and $\mathbf{z}$ such that $\mathbf{w}X^\intercal = \mathbf{z}Y^\intercal$ since $R(X, Y) = 0$. Hence, the last f rows of matrix on the left are linearly independent from the first d rows. Therefore, it is hidden that they share $\mathbf{h}$.

$$\begin{pmatrix} X^\intercal \\ Y^\intercal \end{pmatrix} (\mathbf{h}^\intercal) = \begin{pmatrix} X^\intercal \mathbf{h}^\intercal \\ Y^\intercal \mathbf{h}^\intercal \end{pmatrix}$$