Functional Encryption for Computational Hiding in Prime Order Groups via Pair Encodings

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

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

Abstract. Lewko and Waters introduced the *computational hiding* technique in Crypto'12. In their technique, two computational assumptions that achieve selective and co-selective security proofs lead to adaptive security of an encryption scheme. Later, pair encoding framework was introduced by Attrapadung in Eurocrypt'14. The pair encoding framework generalises the computational hiding technique for functional encryption (FE). It has been used to achieve a number of new FE schemes such as FE for regular languages and unbounded attribute based encryption allowing multi-use of attributes. Nevertheless, the generalised construction of Attrapadung's pair encoding for those schemes is adaptively secure only in composite order groups, which leads to efficiency loss. It remains a challenging task to explore constructions in prime order groups for gaining efficiency improvement, which leaves the research gap in the existing literature. In this work, we aim to address this drawback by proposing a new generalised construction for pair encodings in prime order groups. Our construction will lead to a number of new FE schemes in prime order groups, which have been previously introduced only in composite order groups by Attrapadung.

1 Introduction

Functional Encryption (FE) is rooted in Identity Based Encryption (IBE) [26, 6, 11]. It achieves fine-grained access control based on various types of functions using public keys. Subsequently, more complicated functions, such as boolean functions [24, 13, 18], inner products [16, 21], arithmetic formulas [7, 15] and deterministic finite automata [28] are embedded into FE to support complex systems. In FE, a user does not own a unique key. Instead, multiple key elements are created based on the user's properties. Therefore, keys which the simulator can query are not well separated from keys which it cannot when it sets up the system. This makes security analysis of an FE using traditional partitioning technique [5, 8, 6] become daunting.

Waters introduced dual system encryption [27] methodology as a break-through technique. Using this methodology, a number of FE schemes which are secure even against the adaptive adversary were introduced. In the dual system encryption, all keys and ciphertexts have two types, namely normal and semi-functional, and proving the invariance between both types of a key is a most critical part. In this proof, a semi-functional key correlates with the ciphertext in order to hide its type to the simulator such that the simulator cannot distinguish it from a normal key, but this correlation must be hidden to the adversary. Originally, Waters solved this paradox using information theoretic arguments. Therefore, Waters showed that the correlation can be hidden perfectly to the adversary.

More recently, Lewko and Waters showed that this problem was also solvable using two selective security proofs. Particularly, in their technique hiding correlation between a normal key and a semi-functional key is reduced to selective and co-selective proofs based on traditional partitioning techniques which rely usually on computational assumptions [3]. This *computational hiding* allows

[3] Since the invariance is proved under computational assumptions, the technique is called *computational hiding* [2, 29]. It is also called *doubly selective security* in [2, 4] based on the fact that computational assumptions are originated from their selective and co-selective security analyses.

Table 1. Comparison of Encoding Frameworks

	Composite Order Groups	Prime Order Groups
Perfect hiding[†]	Wee [29], CGW [9]	CGW [9], AC [1]
Computational hiding[‡]	Attrapadung [2]	**Ours** (as well as [3])

[†]If the correlations are hidden by information theoretic arguments as Waters' dual system encryption, it is known as *perfect hiding* since the correlation is hidden perfectly.
[‡]Encodings for computational hiding are also applicable to perfect hiding when computational assumptions can be replaced by information theoretic arguments such as pair-wise independence.

FE schemes to have more desirable properties such as an unbounded universe of attributes [2] and multi-use of attributes [20] for adaptively secure ABE scheme.

Recently, Attrapadung [2] introduced pair encoding framework. As a part of the framework, Attrapadung generalised the computational hiding technique [20] well using encodings and showed that the framework was comprehensive enough to capture many FE schemes. The number of adaptively secure FE schemes such as FE for regular languages and unbounded KP-ABE allowing multi-use of attributes were introduced in composite order groups by the computational hiding technique via pair encodings.

There exist more encoding works given by Wee in [29], Agrawal and Chase in [1] Chen, Gay and Wee in [9]. They more focused on *perfect hiding* of the key invariance the same as Waters' dual system encryption. It is not clear how these works are applicable to FE schemes requiring computational hiding technique. Therefore, Attrapadung's work remains the only encoding which employs the computational hiding. This implies that the generic construction of pair encodings for computational hiding technique has only been proven in composite order groups. Since composite order groups lead to a huge efficiency loss [20, 12, 9], achieving FE in prime order groups efficiently still remains a challenging task.

1.1 Our Contribution

Pair encodings in prime order groups. We introduce a new construction in prime order groups for pair encoding schemes. We adopt the syntax of Attrapadung's pair encoding framework [2] to show that our construction is widely applicable to FE schemes even if they require computational hiding. We provide Table 1 to summarise our construction in comparison to the other constructions for the encodings.

Recently, Attrapadung and Yamada [4] showed that the pair encoding framework preserves duality. Therefore, if one scheme is a pair encoding scheme, its dual scheme is also a pair encoding scheme where the dual scheme is a scheme swapping the predicates between key and ciphertext from the original scheme (e.g. Key Policy Attribute Based Encrption (KP-ABE) and Ciphertext Policy Attribute Based Encryption (CP-ABE)). Based on their finding, pair encoding schemes from [20, 2] were extended to dual schemes and dual policy schemes where a dual policy scheme in which both keys and ciphertexts have their own policy (e.g. A conjugated scheme of KP-ABE and CP-ABE). This enables us to apply our construction to a more wide range of FE schemes.

New Schemes in Prime Order Groups. With our construction, we feature pair encoding schemes [20, 2, 4] which were realised only in composite order groups into prime order groups. We provide the list of FE schemes achievable via our construction in Table 2. Our construction achieves adaptive security of following functional encryption schemes in prime order groups for the first time. They include Doubly Spatial Encryption, FE for regular language, KP-ABE with short ciphertexts, CP-ABE with short keys and unbounded ABE supporting multi-use of attributes, to the best of our knowledge[4].

[4] We note that at present, there is a concurrent work for this claim. We shall describe it in Section 1.2.

Table 2. The comparison between previous works and ours

FEs		Policy			Multi -use	Universe	Order
		CT	Key	Dual			
Doubly Spatial Encryption	A [2]		√		N/A	N/A	Composite
	AY [4]	√		√	N/A	N/A	Composite
	Ours	√	√	√	N/A	N/A	Prime
FE for regular languages	A [2]	√	√		N/A	Large	Composite
	AY [4]			√	N/A	Large	Composite
	Ours	√	√	√	N/A	Large	Prime
Unbounded ABE	OT [23]	√	√		No	Large	Prime
	A [2]		√		Yes	Large	Composite
	AY [4]	√		√	Yes	Large	Composite
	Ours	√	√	√	Yes	Large	Prime
ABE in a small universe	LOSTW [18]	√	√		No	Small	Composite
	A [2]	√	√		No	Small	Composite
	LW [20]	√			Yes	Small	Composite
		√			Yes	Small	Prime
	AY [4]		√	√	Yes	Small	Composite
	Ours	√	√	√	Yes	Small	Prime
ABE with short ciphertexts	A [2]		√		Yes	Large	Composite
	Ours		√		Yes	Large	Prime
ABE with short keys	AY [4]	√			Yes	Large	Composite
	Ours	√			Yes	Large	Prime

Unbounded ABE implies ABE schemes supporting a large universe of attributes and unbounded sized policy. *No* in the column of Multi-use implies that each predicate (e.g. an attribute) can appear only once in a policy.

1.2 Concurrent Work: Comparison with [3]

There is an independent work by Attrapadung [3]. We note that the underlying technique such as the assumptions and the way to achieve important properties required by the dual system encryption is fairly different. Technically, this work utilises the dual system groups [10] as the other encodings [9, 1] do in prime order groups. We do not use the dual system groups for our construction in prime order group. Due to this, although Attrapadung's construction in [3] may offer tighter security using a self-reducible assumption, our construction offers smaller public parameters in comparison to [3] .

1.3 Our Technique

Modular Approach for Functional Encryption. We utilise the pair encoding framework to describe our construction. However, we observed that the existing properties of pair encodings in [2] are insufficient to construct adaptively secure FE in prime order groups. Therefore, we additionally define a new property, namely *linearity over common parameters*. To support this property, we refine the notation of the pair encoding framework. In our notation, common parameters are denoted as (1,$\mathbf{h}$) where $\mathbf{h}$ is a vector of common parameters of the previous notation of pair encodings. It should be noted that this additional property does not harm the generality of the pair encoding framework because, by definition, the pair encoding framework already requires a linear structure over common parameter $\mathbf{h}$.

Nested Dual System Encryption. In Waters' dual system encryption [27], ciphertexts and private keys have two types which are normal and semi-functional (SF). Using these types, proving adaptive security of FE is divided into several relatively easy problems such as proving the invariance between a normal key and a semi-functional key. Later, Lewko and Waters [20] and Attrapadung [2] separated this key invariance problem more specifically to utilise the computational hiding. They additionally define a Nominally Semi-functional (NSF) key and a Temporary

Semi-functional (TSF) key and change a key from normal to SF in sequence of

$$\text{Normal key} \xrightarrow{(Step\,1)} \text{NSF key} \xrightarrow{\text{(computational assump.)}} \text{TSF key} \xrightarrow{(Step\,2)} \text{SF key.}$$

Then, they reduce the invariance between NSF and TSF to computational assumptions such as q-type assumptions. To apply computational assumptions, the semi-functional elements of NSF and TSF keys must be uncorrelated with their normal elements as well as the other private keys. This independence is achieved by the Chinese Remainder Theorem in a composite order group, relatively easily. Because each subgroup is based on different prime orders in a composite order group, the values in one subgroup do not correlate to those in the other subgroups. Therefore, in steps 1 and 2 in the above transition, multiple random values on a subgroup are projected into another subgroup without correlation by a single transition.

In prime order groups, achieving adaptive security is more difficult since there is no obvious separation such as different subgroups to feature semi-functional elements. A technique featuring composite order groups into prime order groups using the Dual Pairing Vector Space (DPVS) [21, 22] was introduced by Lewko [17]. However, DPVS still retains an inefficiency caused by the size of vectors. Also, the size of parameters in [17] increases linearly with the number of parameters to hide. Another approach is to utilise (prime order) dual system groups [10], which enumerates a composite order group element with several elements in prime order groups. This also expands the public parameter size by a constant factor. There are constructions [1, 9] in prime order groups, but it is unsure how they can be applied to schemes requiring computational hiding.

We do not use either DPVS or dual system groups for our construction. Instead, we divide the transitions of a key (steps 1 and 2) into more steps by each random value. We additionally define an NE_j key and a TE_j key to prove the key invariance. Those keys are special cases of NSF keys and TSF keys. An NE_j key is identical to an NSF key having $(r_1, ..., r_j, 0, ..., 0)$ as a randomness. That is, the first j random values are selected randomly, but other random values are being set to 0. A TE_j key is defined similarly to an NE_j key, but using a TSF key. Using these types, we additionally localise each random value within the key and project it as semi-functional elements. For example, in step 1, we prove the invariance of NE_{k-1} key and NE_k to show the invariance of a normal key to an NSF key. Also, we change a TSF key to an SF key using TE_j keys. If we denote m_r as the number of random values in a key, a normal key and a TSF key change to an NSF key and an SF key, resp.

$$Step\,1: \text{ Normal key} \rightarrow NE_1 \rightarrow ... \rightarrow NE_{m_r-1} \rightarrow \text{NSF key}$$

$$Step\,2: \text{ TSF key} \rightarrow TE_{m_r-1} \rightarrow ... \rightarrow TE_1 \rightarrow \text{SF key.}$$

Lewko and Waters' IBE. Since we use the nested dual system encryption, the key invariance problem can be broken down into simpler problems by each random value. To solve those simplified problems, we utilise the assumptions introduced in Lewko and Waters' IBE [19]. Primarily, in their IBE, the technique to provide the key invariance between a normal key and a semi-functional key is only applicable if FEs have one random value to randomise their function parts. It works well for IBE because IBE is one of the simplest forms of FE. However, the way of applying their technique to FEs having more complex structures (i.e. multiple random values for function parts and/or computational hiding) is unknown. In our work, since we successfully simplify the key invariance of pair encodings as we previously described in prime order groups, we can apply their technique to pair encoding framework. Informally, to prove the invariance between NE_i key and NE_{i+1} in our nested model, we consider NE_i key and NE_{i+1} key as a normal key and an SF key of Lewko and Waters' IBE scheme. Except the i^{th} random value the other random values are generated randomly or fixed as 0, then we can fit our problem to the semi-functional key invariance of their IBE.

2 Related Works

A duality of the pair encoding framework was investigated by Attrapadung and Yamada [4]. It means that a symmetric conversion of a pair encoding scheme is also adaptively secure. Since

the converted scheme itself is also a pair encoding scheme, they are also automatically adaptively secure under the pair encoding framework.

Converting encryption schemes in composite order groups into prime order groups [12, 14, 25, 17, 20] have been actively studied in the recent years. Those techniques provide both project and orthogonal properties in prime order groups to feature subgroups of composite order groups. Nevertheless, the techniques of [12, 14, 25] do not hide the values which are projected in prime order groups. This makes applying those techniques to the dual system encryption become difficult. Furthermore, the Dual Pairing Vector Spaces (DPVS) [21, 22] was also introduced. It is a well-known tool to ease the inefficiency caused by composite order groups. A number of FE schemes [18, 20–23, 20] are constructed in prime order groups using DPVS. Unfortunately, DPVS still has inefficiency which is caused by the size of vectors. For example, Lewko [17] introduced a conversion technique to feature FE in composite order group into prime order groups using DPVS, but it has an inherent loss in efficiency. In this technique, the size of vector increases linearly with the number of parameters to hide in the dual system encryption.

Wee [29] introduced the predicate encoding framework. This framework is quite similar to the pair encoding framework, but it is only for perfect hiding. This restriction limits the use of this encoding to relatively simple structures. More recently, Agrawal and Chase [1] and Chen, Gay and Wee [9] introduced a construction in prime order groups for the encodings, independently. However, it is still unclear how to construct FE schemes in prime order groups using the encoding if the schemes require computational hiding.

3 Background

For our construction, we use asymmetric bilinear groups. Let $\mathcal{G}$ be group generator taking a security parameter λ as input and outputting a description of asymmetric bilinear groups G_1, G_2 and G_T. For our purposes, we will have $\mathcal{G}$ output (p, G_1, G_2, G_T, e) where p is a prime, G_1, G_2 and G_T are cyclic groups of order p, and $e : G_1 \times G_2 \to G_T$ which is efficiently computable and non-degenerate. Also, the group descriptions of G_1, G_2 and G_T include generators of the respective cyclic groups.

3.1 Complexity Assumptions

For our security analysis, we introduce following assumptions which originally appear in the full version of [19]. Given a group generator $\mathcal{G}$, a description of bilinear groups (p, G_1, G_2, G_T, e) is created randomly from $\mathcal{G}$.

Assumption 1 (*LW1*) Let $f_1 \in G_1$ and $f_2 \in G_2$ be chosen randomly. Let $a, c, d \in \mathbb{Z}_p$ be selected randomly. Given

$$D := \{f_1, f_1^a, f_1^{ac^2}, f_1^c, f_1^{c^2}, f_1^{c^3}, f_1^d, f_1^{ad}, f_1^{cd}, f_1^{c^2 d}, f_1^{c^3 d} \in G_1, f_2, f_2^c \in G_2\},$$

it is hard to distinguish between $T_0 = f_1^{ac^2 d}$ and $T_1 \xleftarrow{R} G_1$.

We define the advantage of the adversary $\mathcal{A}$ to break assumption 1 as

$$Adv_{\mathcal{A}}^{LW1}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|.$$

Assumption 2 (*LW2*) Let $f_1 \in G_1$ and $f_2 \in G_2$ be chosen randomly. Let $c, d, t, w \in \mathbb{Z}_p$ be selected randomly. Given

$$D := \{f_1, f_1^d, f_1^{d^2}, f_1^{tw}, f_1^{dtw}, f_1^{d^2 t} \in G_1, f_2, f_2^c, f_2^d, f_2^w \in G_2\},$$

it is hard to distinguish between $T_0 = f_2^{cw}$ and $T_1 \xleftarrow{R} G_2$.

We define the advantage of the adversary $\mathcal{A}$ to break assumption 2 as

$$Adv_{\mathcal{A}}^{LW2}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|.$$

Assumption 3 (*LW3*) Let $f_1 \in G_1$ and $f_2 \in G_2$ be chosen randomly. Let $a, b, c \in \mathbb{Z}_p$ be selected randomly. Given

$$D := \{f_1, f_1^a, f_1^c, f_1^d \in G_1, f_2, f_2^a, f_2^c, f_2^d \in G_2\},$$

it is hard to distinguish between $T_0 = e(f_1, f_2)^{acd}$ and $T_1 \xleftarrow{R} G_T$.

We define the advantage of the adversary $\mathcal{A}$ to break assumption 3 as

$$Adv_{\mathcal{A}}^{LW3}(\lambda) = |\Pr[\mathcal{A}(D, T_0) = 1] - \Pr[\mathcal{A}(D, T_1) = 1]|.$$

In addition to assumptions 1, 2 and 3, our framework requires computational assumptions to prove the computational hiding properties of pair encoding schemes. Computational hiding of a pair encoding is usually proved using q-type assumptions [2]. For example, to achieve functional encryption for regular languages, our framework requires two q-type assumptions (ℓ-EDHE1 Assumption and (n, m)-EDHE2 Assumption) from [2].

3.2 Functional Encryption

We adopt the definition of functional encryption and its adaptive security of [2].

Definition of Functional Encryption [2]. A functional encryption for a function R consists of Setup, Encrypt, KeyGen and Decrypt as follows:

Setup $(1^\lambda, \kappa) \to (PK, MSK)$: The algorithm takes in a security parameter 1^λ and an index κ which is allocated uniquely for the function R. It outputs a public parameter PK and a master secret key MSK.

Encrypt $(x, M, PK) \to CT$: The algorithm takes in a predicate $x \in \mathcal{X}$, a public parameter PK and a plaintext M. It outputs a ciphertext CT.

KeyGen $(y, MSK, PK) \to SK$: The algorithm takes in a predicate $y \in \mathcal{Y}$, a master secret key MSK and a public parameter PK. It outputs a private key SK.

Decrypt $(PK, SK, CT) \to M$: the algorithm takes in a key SK for y and a ciphertext CT for x. If $R(x, y) = 1$, it outputs a message $M \in \mathcal{M}$. Otherwise, it aborts.

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

Definition of Adaptive Security of Functional Encryption [2]. A functional encryption for a function 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 challenge $\mathcal{C}$ defined below.

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

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

Challenge : When the adversary requests the challenge ciphertext of $x \in \mathcal{X}$, for $R(x, y_i) = 0$; $\forall i \in [1, q_1]$, and submits two messages M_0 and M_1, the challenger randomly selects b from $\{0, 1\}$ and returns the challenge ciphertext CT created by running $\mathsf{Encrypt}(x, M_b, PK)$.

Phase 2 : This is identical with Phase 1 except for the additional restriction that $y_i \in \mathcal{Y}$ for $i \in [q_1 + 1, q_t]$ such that $R(x, y_i) = 0$; $\forall i \in [q_1 + 1, q_t]$

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

We define an adversary $\mathcal{A}$'s advantage as $Adv_{\mathcal{A}}^{FE}(\lambda) := |\Pr[b = b'] - 1/2|$.

4 Overview of the Pair Encoding Framework

In this section, the syntax and properties of the pair encoding framework [2] are briefly provided. Additionally, we define a new property, *linearity over common parameters*. This property means that two encodings are linear in common parameters if they have the same random values and predicates. To support the property, we refine the notation of pair encoding framework. In our notation common parameters are notated as $(1, \mathbf{h})$, where $\mathbf{h}$ is a vector of common parameters of previous notation of pair encodings. By including the constant as common parameters, we can describe our new property, more clearly.

4.1 Syntax of Pair Encodings [2]

For a function $R : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ and a prime p, a pair encoding $P(R,p)$ consists of four deterministic algorithms (Param, Enc1, Enc2, Pair).

Param $(\kappa) \to (\mathcal{H}, \mathcal{R}_r, \mathcal{R}_s)$: An index κ for a predicate family is taken as input. The algorithm outputs domains for Enc1 and Enc2. Hence, it set up the domain of common parameters $\mathbf{h}$ as $\mathcal{H}$. Also, it sets up $\mathcal{R}_r$, $\mathcal{R}_s$ which are domains of random values $\mathbf{r}$ and $\mathbf{s}$, respectively. It outputs the descriptions of $\mathcal{H}, \mathcal{R}_r, \mathcal{R}_s$.

Enc1 $(\alpha, x, \mathbf{h}, \mathbf{r}) \to \mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})$: It takes $(\alpha, x, \mathbf{h}, \mathbf{r}) \in \mathbb{Z}_p \times \mathcal{X} \times \mathcal{H} \times \mathcal{R}_r$ as inputs and outputs a vector $\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})$. Each coordinate of $\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})$ is a linear combination of monomials α, r_j, $h_i r_j$ where $\mathbf{r} = (r_1, ..., r_{m_r})$ and $\mathbf{h} = (h_1, ..., h_n)$.

Enc2 $(y, \mathbf{h}, s, \mathbf{s}) \to \mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})$: It takes $(y, \mathbf{h}) \in \mathcal{Y} \times \mathcal{H}$ and randomness $s \in \mathbb{Z}_p$ and $\mathbf{s} \in \mathcal{R}_s$ as inputs and outputs a vector $\mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})$. Each coordinate of $\mathbf{c}(y, (1,\mathbf{h}); \mathbf{r})$ is a linear combination of monomials s, s_j, $h_i s_j$, where $\mathbf{s} = (s_1, ..., s_{m_s})$ and $\mathbf{h} = (h_1, ..., h_n)$.

Pair $(x, y) \to M_{xy}$: It takes, as input, predicates x and y and outputs reconstruction matrix M_{xy}.

Correctness. We let $m_1 = |\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})|$ and $m_2 = |\mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})|$. If $R(x,y) = 1$, Pair outputs an $m_1 \times m_2$ reconstruction matrix M_{xy} such that

$$\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r}) M_{xy} \mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})^\intercal = \sum_{i \in [1,m_1], j \in [1,m_2]} k_i M_{xy_{i,j}} c_j = \alpha s$$

where k_i and c_j are the i^{th} coordinate of $\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})$ and the j^{th} coordinate of $\mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})$, resp.

In our construction, the output of Enc1 is used for private keys, and Enc2 is utilised for ciphertexts. Also, a reconstruction matrix means that there exists a proper decryption algorithm. Intuitively, $\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})$ and $\mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})$ are considered as the exponents of a private key and a ciphertext and the result of the reconstruction, αs, is an exponent of a group element which hides a message.

All pair encoding schemes must satisfy the properties following:

Property 1. (*Computational α hiding*) For all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R(x,y) = 0$, there exist two polynomially indistinguishable oracles O^1_{AH} and O^2_{AH} as follows:

$O^{\{1,2\}}_{AH}$: When the oracle is initiated by receiving an order between *type k* and *type c* queries, it randomly selects $\mathbf{h} \in \mathcal{H}$ and outputs initial instances $\{f_1, f_2\}$.

When the oracle receives a *type k* query for $x \in \mathcal{X}$, it randomly generates $\mathbf{r} \in \mathcal{R}_r$. If the oracle is O^1_{AH}, it sets $\alpha' = 0$. If the oracle is O^2_{AH}, it sets α' as a random value from $\mathbb{Z}_p$. Then, it returns as a *type k* response $f_2^{\mathbf{k}(\alpha', x, (1,\mathbf{h}); \mathbf{r})}$.

When a *type c* query is received for $y \in \mathcal{Y}$, the oracle randomly generates $\mathbf{s} \in \mathcal{R}_s$ and $s \in \mathbb{Z}_p$ and outputs as a *type c* response $f_1^{\mathbf{c}(y, (1,\mathbf{h}); s, \mathbf{s})}$. It should be noted that the oracles only respond if $R(x,y) = 0$.

Property 2. (*Parameter vanishing*) There exists an element $\mathbf{0} \in \mathcal{R}_r$ such that for all $(\alpha, x, h, \mathbf{h}) \in \mathbb{Z}_p \times \mathcal{X} \times \mathbb{Z}_p \times \mathcal{H}$, $\mathbf{k}(\alpha, x, (h, \mathbf{h}); \mathbf{0})$ is statistically independent of $(h, \mathbf{h})$, that is, for all $h, h' \in \mathbb{Z}_p$ and $\mathbf{h}, \mathbf{h}' \in \mathcal{H}$:

$$\mathbf{k}(\alpha, x, (h, \mathbf{h}); \mathbf{0}) = \mathbf{k}(\alpha, x, (h', \mathbf{h}'); \mathbf{0}).$$

Property 3. (*Linearity over random values*) For all $(x, h, \mathbf{h}) \in \mathcal{X} \times \mathbb{Z}_p \times \mathcal{H}$, $\mathbf{k}(\cdot, x, (h, \mathbf{h}); \cdot)$ is linear in α and $\mathbf{r}$. Also, for all $(y, h, \mathbf{h}) \in \mathcal{Y} \times \mathbb{Z}_p \times \mathcal{H}$, $\mathbf{c}(y, (h, \mathbf{h}); \cdot, \cdot)$ is linear in s and $\mathbf{s}'$.

Property 4. (*Linearity over common parameters*) For all $(x, \mathbf{r}) \in \mathcal{X} \times \mathcal{R}_r$, $\mathbf{k}(\cdot, x, (\cdot, \cdot); \mathbf{r})$ is linear in α, h and $\mathbf{h}$. Also, for all $(y, s, \mathbf{s}) \in \mathcal{X} \times \mathbb{Z}_p \times \mathcal{R}_s$, $\mathbf{c}(y, (\cdot, \cdot); s, \mathbf{s})$ is linear in h and $\mathbf{h}$.

Remark 1. We additionally define *linearity over common parameters* for pair encodings, but this does not harm the generality of the pair encoding. The pair encoding framework is not only defined by properties. In [2], the output of $\mathsf{Enc1}$ must be linear combinations of monomials α, r_i and $h_j r_i$ where h_j is the j^{th} coordinate of $\mathbf{h}$ and r_i is the i^{th} coordinate of $\mathbf{r}$. Therefore, with the fixed $\mathbf{r}$, the linearity over common values trivially holds, by denoting the coefficient of a monomial r_i as 1 and including it as common values. Also, the output of $\mathsf{Enc2}$ which is a linear combination of s, s_i, $h_j s$ and $h_j s_i$ also satisfies this property as the same reason where s_i is the i^{th} coordinate of $\mathbf{s}$.

4.2 Computational α hiding

To prove *computational* α *hiding*, we additionally define *selective* α *hiding* and *co-selective* α *hiding* by the order of queries. First, we define oracles O^1_{SAH}, O^2_{SAH}, O^1_{CAH} and O^2_{CAH} to describe this notion. Each oracle responds for a *type* k query by using an output of $\mathsf{Enc1}$, and a *type* c query for using an output of $\mathsf{Enc2}$ for (x, y) such that $R(x, y) = 0$.

$O^{\{1,2\}}_{SAH}$: These oracles are identical with $O^{\{1,2\}}_{AH}$ except the order of queries. For selective α hiding, the oracles work only if a *type* c query is requested in advance of a *type* k query.
$O^{\{1,2\}}_{CAH}$: These oracles are identical with $O^{\{1,2\}}_{AH}$ except the order of queries. For co-selective α hiding, the oracles work only if a *type* k query is requested in advance of a *type* c query.

Selective α hiding We define the advantage of the adversary $\mathcal{A}$ to distinguish between O^1_{SAH} and O^2_{SAH} as

$$Adv^{O_{SAH}}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(O^1_{SAH}) = 1] - \Pr[\mathcal{A}(O^2_{SAH}) = 1]|.$$

Co-selective α hiding We define the advantage of the adversary $\mathcal{A}$ to distinguish between O^1_{CAH} and O^2_{CAH} as

$$Adv^{O_{CAH}}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(O^1_{CAH}) = 1] - \Pr[\mathcal{A}(O^2_{CAH}) = 1]|.$$

5 Construction

For a pair encoding $P(R, p)$, a functional encryption $FE(P)$ comprises four randomised algorithms employing prime order groups G_1, G_2 and G_T of order p. In the construction, we use the subscripts of group elements to denote the group generators to be used to generate those elements (e.g. $g_1, f_1 \in G_1$ and $g_2, f_2 \in G_2$).

$\mathsf{Setup}(\lambda) \rightarrow PK, MSK$: The setup algorithm selects a bilinear group G_1, G_2, G_T of order p. The algorithm randomly selects $g_1 \in G_1$ and $g_2 \in G_2$. Then, it runs $\mathsf{Param}(\kappa)$ to derive the descriptions of domains for parameters where κ is the index allocated to a predicate family utilising the function R. It randomly generates $\alpha, a, b, y_u, y_v \in \mathbb{Z}_p$ and $\mathbf{h} \in \mathcal{H}$ sets $\tau = y_v + a \cdot y_u$. It publishes public parameters as

$$\{g_1, g_1^{\mathbf{h}}, g_1^a, g_1^{a \cdot \mathbf{h}}, g_1^\tau, g_1^{\tau \cdot \mathbf{h}}, e(g_1, g_2)^\alpha\}.$$

It also sets MSK as $\{g_2, g_2^\alpha, g_2^{\mathbf{h}}, v_2 = g_2^{b \cdot y_v}, u_2 = g_2^{b \cdot y_u}, f_2 = g_2^b\}$.

Encrypt$(M, y, PK) \to CT_y$: The encryption algorithm randomly chooses $(s, \mathbf{s}) \in \mathbb{Z}_p \times \mathcal{R}_s$ and runs Enc2 to get $\mathbf{c}(y, (1, \mathbf{h}); s, \mathbf{s})$[5]. It sets the ciphertext as

$$C = M \cdot e(g_1, g_2)^{\alpha s}, \mathbf{C}_0 = g_1^{\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, \mathbf{C}_1 = g_1^{a \cdot \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, \mathbf{C}_2 = g_1^{\tau \cdot \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}.$$

KeyGen$(x, MSK, PK) \to SK_x$: The key generation algorithm chooses a random vector $\mathbf{r} \in \mathcal{R}_r$ and runs Enc1 to get $\mathbf{k}(\alpha, x, (1, \mathbf{h}); \mathbf{r})$[6] . Then, it randomly selects $\mathbf{z} \in \mathbb{Z}_p^{k_m}$ where k_m is $|\mathbf{k}|$. Finally, it generates the private key following

$$\mathbf{K}_0 = g_2^{\mathbf{k}(\alpha,x,(1,\mathbf{h});\mathbf{r})} v_2^{\mathbf{z}}, \quad \mathbf{K}_1 = u_2^{\mathbf{z}}, \quad \mathbf{K}_2 = f_2^{-\mathbf{z}}.$$

Decrypt$(PK, x, y, SK_x, CT_y) \to M$ If $R(x, y) = 1$, the algorithm computes a reconstruction matrix M_{xy} such that $\mathbf{k}(\alpha, x, (1, \mathbf{h}); \mathbf{r}) M_{xy} \mathbf{c}(y, (1, \mathbf{h}); \mathbf{s}) = \alpha s$ by Pair. It computes

$$e(g_1, g_2)^{\alpha s} = e(\mathbf{C}_0, (\mathbf{K}_0)^{M_{xy}}) e(\mathbf{C}_1, (\mathbf{K}_1)^{M_{xy}}) e(\mathbf{C}_2, (\mathbf{K}_2)^{M_{xy}}).$$

Finally, the message can be recovered as $C / e(g_1, g_2)^{\alpha s}$.

Correctness We let $A_1 := e(\mathbf{C}_0, (\mathbf{K}_0)^{M_{xy}})$, $A_2 := e(\mathbf{C}_1, (\mathbf{K}_1)^{M_{xy}})$ and $A_3 := e(\mathbf{C}_2, (\mathbf{K}_2)^{M_{xy}})$. Then, we can calculate

$$\begin{aligned}
A_1 &= e(\mathbf{C}_0, (\mathbf{K}_0)^{M_{xy}}) \\
&= e(g_1^{\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, (g_2^{\mathbf{k}(\alpha,x,(1,\mathbf{h});\mathbf{r})} v_2^{\mathbf{z}})^{M_{xy}}) \\
&= e(g_1^{\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, (g_2^{\mathbf{k}(\alpha,x,(1,\mathbf{h});\mathbf{r})})^{M_{xy}}) \cdot e(g_1^{\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, v_2^{\mathbf{z} M_{xy}})) \\
&= e(g_1, g_2)^{\mathbf{k}(\alpha,x,(1,\mathbf{h});\mathbf{r}) M_{xy} \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})} e(g_1, v_2)^{\mathbf{z} M_{xy} \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})} \\
&= e(g_1, g_2)^{\alpha s} \cdot e(g_1, g_2)^{b y_v \mathbf{z} M_{xy} \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}
\end{aligned}$$

$$\begin{aligned}
A_2 &= e(\mathbf{C}_1, (\mathbf{K}_1)^{M_{xy}}) = e(g_1^{a \cdot \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, (u_2^{\mathbf{z}})^{M_{xy}}) \\
&= e(g_1, g_2)^{a b y_u \mathbf{z} M_{xy} \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}
\end{aligned}$$

$$\begin{aligned}
A_3 &= e(\mathbf{C}_2, (\mathbf{K}_2)^{M_{xy}}) = e(g_1^{\tau \cdot \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}, f_2^{-\mathbf{z}}) \\
&= e(g_1, g_2)^{-b(y_v + a \cdot y_u) \mathbf{z} M_{xy} \mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}
\end{aligned}$$

Therefore, the correctness of decryption algorithm holds since

$$e(g_1, g_2)^{\alpha \cdot s} = A_1 \cdot A_2 \cdot A_3.$$

6 Security Analysis

We prove the adaptive security of our construction in the following theorem.

Theorem 1. *Suppose the assumptions* LW1, LW2 *and* LW3 *hold in* $\mathcal{G}$*. For all pair encodings* $P(R, p)$ *where* R *is a predicate function and* p *is a prime, our construction* $FE(P)$ *is adaptive secure. More precisely, for any PPT adversary* $\mathcal{A}$*, there exist PPT algorithms* $\mathcal{B}_1$*,* $\mathcal{B}_2$*,* $\mathcal{B}_3$*,* $\mathcal{B}_4$ *and* $\mathcal{B}_5$*, whose running times are the same as* $\mathcal{A}$*, such that for any* λ*,*

$$Adv_{\mathcal{A}}^{FE(P)}(\lambda) \leq (m_s + 1) \cdot Adv_{\mathcal{B}_1}^{LW1}(\lambda) + 2 \cdot (q_1 + q_2) \cdot m_r \cdot Adv_{\mathcal{B}_2}^{LW2}(\lambda) + Adv_{\mathcal{B}_3}^{LW3}(\lambda)$$

[5] The algorithm only knows the values of x and $\mathbf{r}$. Therefore, $\mathbf{c}(y, (1, \mathbf{h}); s, \mathbf{s})$ is a multivariate linear function of α and $\mathbf{h}$. However, due to the linearity, all elements in a ciphertext can be calculated because $g_1^{\mathbf{h}}, g_1^{a\mathbf{h}}$ and $g_1^{\tau\mathbf{h}}$ are given.

[6] Similar to Encrypt, $\mathbf{K}_0$ can be calculated using g_2^{α} and $g_2^{\mathbf{h}}$

$$+q_1 \cdot Adv_{\mathcal{B}_4}^{O_{CMH}}(\lambda) + q_2 \cdot Adv_{\mathcal{B}_5}^{O_{SMH}}(\lambda)$$

where q_1 and q_2 are the numbers of key queries in Phase I and Phase II, repsectivley, and m_s and m_r are the sizes of randomization parameters used in the encoding $P(R,p)$.

Proof: Theorem 1 is proved by Lemmas 1 to 3. □

To prove adaptive security of our construction, we define several types of keys and ciphertexts, which are only used in security analysis. In following definitions, $(\mathbf{K}'_0, \mathbf{K}'_1, \mathbf{K}'_2)$ and $(C', \mathbf{C}'_0, \mathbf{C}'_1, \mathbf{C}'_2)$ denote a normal key for $x \in \mathcal{X}$ and a ciphertext for $y \in \mathcal{Y}$ which are generated using KeyGen and Encrypt.

Semi-functional (SF) Ciphertext The algorithm randomly selects $(\mathbf{h}', s', \mathbf{s}') \in \mathcal{H} \times \mathbb{Z}_p \times \mathcal{R}_s$. It sets $f_1 = g_1^b$ and $u_1 = g_1^{by_u}$. For a predicate $y \in \mathcal{Y}$, it outputs a semi-functional CT as

$$C = C,\ \mathbf{C}_0 = \mathbf{C}'_0,\ \mathbf{C}_1 = \mathbf{C}'_1 \cdot f_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')},\ \mathbf{C}_2 = \mathbf{C}'_2 \cdot {u_1}^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')}.$$

Semi-functional (SF) Key The algorithm randomly selects $\alpha' \in \mathbb{Z}_p$. For a predicate $x \in \mathcal{X}$, it outputs a semi-functional key as

$$\mathbf{K}_0 = \mathbf{K}'_0 \cdot f_2^{-a\cdot\mathbf{k}(\alpha',x,(0,\mathbf{0});\mathbf{0})},\ \mathbf{K}_1 = \mathbf{K}'_1 \cdot f_2^{\mathbf{k}(\alpha',x,(0,\mathbf{0});\mathbf{0})},\ \mathbf{K}_2 = \mathbf{K}'_2.$$

$\mathbf{E}_j$ Ciphertext The algorithm creates a normal ciphertext $\{C, \mathbf{C}'_0, \mathbf{C}'_1, \mathbf{C}'_2\}$ and selects $s' \in \mathbb{Z}_p$. If $j = 0$, the algorithm set $\mathbf{s}'_0 = \mathbf{0}$. Otherwise, the algorithm randomly selects $s'_1, ..., s'_j$ from $\mathbb{Z}_p$ and it sets $\mathbf{s}'_j = (s'_1, ..., s'_j, 0, ..0)$ and $f_1 = g_1^b$ and $u_1 = g_1^{by_u}$. It outputs the E_j ciphertext as

$$C = C, \quad \mathbf{C}_0 = \mathbf{C}'_0, \quad \mathbf{C}_1 = \mathbf{C}'_1 \cdot f_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}'_j)}, \quad \mathbf{C}_2 = \mathbf{C}'_2 \cdot {u_1}^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}'_j)}$$

It should be noted that E_{m_s} ciphertext is identical to the semi-functional ciphertext by definitions if we write $\mathcal{R}_s = \mathbb{Z}_p^{m_s}$.

$\mathbf{NE}_j$ key The algorithm generates a normal key $(\mathbf{K}'_0, \mathbf{K}'_1, \mathbf{K}'_2)$ using KeyGen. Then it randomly selects $r'_1, ..., r'_j$ from $\mathbb{Z}_p$ and sets $\mathbf{r}'_j = (r'_1, ..., r'_j, 0, ..0) \in \mathcal{R}_r$. Finally, it randomly generates $\mathbf{h}'$ and outputs an NE_j key as

$$\mathbf{K}_0 = \mathbf{K}'_0 \cdot f_2^{-a\cdot\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_j)}, \quad \mathbf{K}_1 = \mathbf{K}'_1 \cdot f_2^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_j)}, \quad \mathbf{K}_2 = \mathbf{K}'_2.$$

An NE_0 key is identical to a normal key since $\mathbf{k}(0, x, (1, \mathbf{h}'); (0, ..., 0)) = \mathbf{0}$ by the definition of pair encodings. Additionally, we call NE_{m_r} key as a Nominally Semi-functional (NSF) Key if we write $\mathcal{R}_r = \mathbb{Z}_p^{m_r}$.

$\mathbf{TE}_j$ Key The algorithm generates a normal key $(\mathbf{K}'_0, \mathbf{K}'_1, \mathbf{K}'_2)$ using KeyGen. Then, it randomly selects $\alpha' \in \mathbb{Z}_p$, $r'_1, ..., r'_j \in \mathbb{Z}_p$ and $\mathbf{h}' \in \mathcal{H}$. Finally, it sets $\mathbf{r}'_j = (r'_1, ..., r'_j, 0, ..0) \in \mathcal{R}_r$ and outputs a TE_j key as

$$\mathbf{K}_0 = \mathbf{K}'_0 \cdot f_2^{-a\cdot\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{r}'_j)}, \quad \mathbf{K}_1 = \mathbf{K}'_1 \cdot f_2^{\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{r}'_j)}, \quad \mathbf{K}_2 = \mathbf{K}'_2.$$

A TE_0 key is identical to an SF key since $\mathbf{k}(\alpha', x, (1, \mathbf{h}'); \mathbf{0}) = \mathbf{k}(\alpha', x, (0, \mathbf{0}); \mathbf{0})$ by *parameter vanishing*. Additionally, we call TE_{m_r} key as a Temporary Semi-functional (TSF) Key if we write $\mathcal{R}_r = \mathbb{Z}_p^{m_r}$.

All NE_j key including an NSF key can decrypt both normal and semi-functional ciphertexts by sharing $\mathbf{h}'$ in semi-functional elements with semi-functional ciphertexts. However, TE_j keys only decrypt normal ciphertexts. If they decrypt semi-functional ciphertexts, $e(f_1, f_2)^{\alpha' s}$ prevents the decryption even if they share $\mathbf{h}'$ with semi-functional ciphertexts.

We define security games consisting of different types of keys and ciphertexts. We will prove that all games are indistinguishable through our analysis.

Table 3. The summary of semi-functional ciphertext invariance

Exponents of semi-functional elements of the chanllenge CT for y	Type of the challenge CT	Games	Remarks
$\mathbf{c}(y,(1,\mathbf{h}');0,(0,...,0))$	Normal	Game_{Real}	$\mathsf{Game}_{Real} \approx \mathsf{Game}_{Real,0}$
$\mathbf{c}(y,(1,\mathbf{h}');\boxed{s'},(0,...,0))$	E_0	$\mathsf{Game}_{Real,0}$	by Lemma 1.1
$\mathbf{c}(y,(1,\mathbf{h}');s',(\boxed{s'_1},0,0,...,0))$	E_1	$\mathsf{Game}_{Real,1}$	$\mathsf{Game}_{Real,i} \approx$
...	...	...	$\mathsf{Game}_{Real,i+1}$
$\mathbf{c}(y,(1,\mathbf{h}');s',(s'_1,s'_2,s'_3...,\boxed{s'_{m_s}}))$	NSF ($=\mathrm{E}_{m_s}$)	Game_0 *	by Lemma 1.2

*:Game_0 is equal to Game_{Real,m_s} by the definitions.

Game_{Real} This game is identical with the adaptive security model. All keys and the challenge ciphertext are normal in this game.

Game_i^N This game is identical with Game_{Real} except the types of the first i keys and the challenge ciphertext. In this game, the first $i-1$ keys and the challenge ciphertext are semi-functional and the i^{th} key is an NSF key.

Game_i^T This game is identical with Game_i^N except the type of the i^{th} key. In this game, the i^{th} key is a TSF key.

Game_i This game is identical with Game_i^T except the type of the i^{th} key. In this game, the i^{th} key is semi-functional.

Game_{Final} This game is identical with Game_{q_t} except the message encrypted in the challenge ciphertext where q_t is a total number of queries that are queried by the adversary. A random message replaces the encrypted message.

6.1 Semi-functional Ciphertext Invariance

First, to prove the invariance between Game_{Real} and Game_0 (i.e. semi-functional ciphertext invariance), we additionally define $\mathsf{Game}_{Real,j}$ using E_j ciphertext for all $j \in [0, m_s]$ if we write $\mathcal{R}_s = \mathbb{Z}_p^{m_s}$.

$\mathsf{Game}_{Real,j}$ This game is identical with Game_{Real} except the type of the challenge ciphertext. In this game, the challenge ciphertext is E_j. It should be noted that Game_{Real,m_s} is identical with Game_0 since a E_{m_s} ciphertext is identical with a semi-functional ciphertext by the definitions.

We first show that Game_{Real} and $\mathsf{Game}_{Real,0}$ are invariant in Lemma 1.1. Then, in Lemma 1.2, it is proved that $\mathsf{Game}_{Real,i-1}$ and $\mathsf{Game}_{Real,i}$ is indistinguishable for all $i \in [1, m_s]$. We provide Table 3 to summarize these processes.

Lemma 1. Suppose there exists a PPT adversary who distinguishes Game_{Real} and Game_0 with a non-negligible advantage ϵ, then an algorithm which breaks $LW1$ can be built with the advantage ϵ using the adversary.

Proof: This lemma is proved by Lemmas 1.1. and 1.2. □

Lemma 1.1. Suppose there exists a PPT adversary $\mathcal{A}$ who distinguishes Game_{Real} and $\mathsf{Game}_{Real,0}$ with a non-negligible advantage ϵ, then an algorithm which breaks $LW1$ can be built with the advantage ϵ using $\mathcal{A}$.

Proof: In this proof, we will show that the hardness to distinguish Game_{Real} and $\mathsf{Game}_{Real,0}$ is reduced to $LW1$ assumption. Therefore, using the given instance from $LW1$

$$\{f_1, f_1^a, f_1^{ac^2}, f_1^c, f_1^{c^2}, f_1^{c^3}, f_1^d, f_1^{ad}, f_1^{cd}, f_1^{c^2d}, f_1^{c^3d}, T \in G_1, f_2, f_2^c \in G_2\},$$

$\mathcal{B}$ will simulate Game_{Real} and $\mathsf{Game}_{Real,0}$ depending on the value of T using $\mathcal{A}$ to break the assumption.

Setup: $\mathcal{B}$ randomly selects $\alpha, y_g, y_u \in \mathbb{Z}_p, \mathbf{h}', \mathbf{h}'' \in \mathbb{Z}_p^n$ and it sets

$$g_1 = f_1^{c^2} f_1^{y_g}, g_1^{\mathbf{h}} = (f_1^{c^2})^{\mathbf{h}'} f_1^{\mathbf{h}''}, g_1^a = f_1^{ac^2} (f_1^a)^{y_g}, g_1^{a\cdot\mathbf{h}} = (f_1^{ac^2})^{\mathbf{h}'} (f_1^a)^{\mathbf{h}''}$$

$$g_2^{\alpha} = (f_2^{c^2})^{\alpha} f_2^{\alpha y_g}, g_2 = f_2^{c^2} f_2^{y_g}, g_2^{\mathbf{h}} = (f_2^{c^2})^{\mathbf{h}'} f_2^{\mathbf{h}''}, g_2^{b} = f_2, v_2 = f_2^{c}, u_2 = f_2^{y_u}.$$

The values of $\mathbf{h}$ are set implicitly since the simulator does not know the value of c^2. Also, it implies that $\tau = c + ay_u$, $y_v = c$ and $b^{-1} = c^2 + y_g$. It publishes public parameters as follows:

$$g_1, g_1^{a}, g_1^{a\cdot\mathbf{h}}, g_1^{\tau} = f_1^{c^3}(f_1^{ac^2})^{y_u}(f_1^{c})^{y_g}(f_1^{a})^{y_u y_g}, g_1^{\tau\cdot\mathbf{h}} = (f_1^{c^3})^{\mathbf{h}'}(f_1^{ac^2})^{y_u\mathbf{h}'}(f_1^{c})^{\mathbf{h}''}(f_1^{a})^{y_u\mathbf{h}''}$$

$$e(g_1, g_2)^{\alpha} = e(f_1^{c^3}, f_2^{c})^{\alpha} e(f_1^{c^2}, f_2)^{2\alpha\cdot y_g} e(f_1, f_2)^{\alpha\cdot y_g^2},$$

Phase I and II: $\mathcal{B}$ randomly selects $\mathbf{z}' \in \mathbb{Z}_p^{m_k}$ and $\mathbf{r} \in \mathcal{R}_r$ where $m_k = |\mathbf{k}(\alpha, x, (1, \mathbf{h}'); \mathbf{r})|$. Then, it implicitly sets

$$\mathbf{z} = \mathbf{z}' - \mathbf{k}(\alpha c, x, (c, c\cdot\mathbf{h}'); \mathbf{r}).$$

$\mathbf{z}$ is randomly distributed due to $\mathbf{z}'$. To create normal keys, $\mathcal{B}$ sets

$$\mathbf{K}_0 = f_2^{\mathbf{k}(\alpha y_g, x, (y_g, \mathbf{h}''); \mathbf{r})} \cdot (f_2^{c})^{\mathbf{z}'}$$

$$\mathbf{K}_1 = f_2^{y_u\mathbf{z}'}(f_2^{c})^{-y_u\mathbf{k}(\alpha, x, (1,\mathbf{h}'), \mathbf{r})}, \mathbf{K}_2 = f_2^{-\mathbf{z}'}(f_2^{c})^{\mathbf{k}(\alpha, x, (1,\mathbf{h}'), \mathbf{r})}$$

Using f_2^c, $\mathbf{K}_0, \mathbf{K}_1$ and $\mathbf{K}_2$ can be calculated. $\mathbf{K}_0$ is properly distributed since

$$\begin{aligned}
\mathbf{K}_0 &= f_2^{\mathbf{k}(\alpha y_g, x, (y_g, \mathbf{h}''); \mathbf{r})} \cdot f_2^{c\mathbf{z}'} \\
&= f_2^{\mathbf{k}(\alpha y_g + \alpha c^2, x, (y_g + c^2, \mathbf{h}'' + c^2\mathbf{h}'); \mathbf{r})} \cdot f_2^{-\mathbf{k}(\alpha c^2, x, (c^2, c^2\mathbf{h}'); \mathbf{r})} f_2^{c\mathbf{z}'} && (1) \\
&= g_2^{\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})} \cdot f_2^{c\mathbf{z}' - c\mathbf{k}(\alpha c, x, (c, c\cdot\mathbf{h}'), \cdot\mathbf{r})} && (2) \\
&= g_2^{\mathbf{k}(\alpha, x, (1,\mathbf{h}); \mathbf{r})} \cdot v_2^{\mathbf{z}}.
\end{aligned}$$

The equalities (1) and (2) hold because of linearity over common parameters.

Challenge: When the adversary asks the challenge ciphertext with messages M_0 and M_1. $\mathcal{B}$ randomly chooses β from $\{0, 1\}$ and $\mathbf{s}$ from $\mathcal{R}_s$ and sets $s = d$. The value of d does not used anywhere else. Therefore, this setting is hidden to the adversary. The algorithm calculates the challenge ciphertexts as follows:

$$C = M_\beta \cdot e(f_1^{c^3 d}, f_2^{c})^{\alpha} e(f_1^{c^2 d}, f_2)^{2\alpha\cdot y_g} e(f_1^{d}, f_2)^{\alpha\cdot y_g^2},$$

$$\mathbf{C}_0 = (f_1^{c^2 d})^{\mathbf{c}(y,(1,\mathbf{h}';);1,0)}(f_1^{c^2})^{\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{s})}(f_1^{d})^{\mathbf{c}(y,(y_g,\mathbf{h}'');1,0)}(f_1)^{\mathbf{c}(y,(y_g,\mathbf{h}'';);0,\mathbf{s})},$$

$$\mathbf{C}_1 = T^{\mathbf{c}(y,(1,\mathbf{h}';);1,0)}(f_1^{ac^2})^{\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{s})}(f_1^{ad})^{\mathbf{c}(y,(y_g,\mathbf{h}'');1,0)}(f_1^{a})^{\mathbf{c}(y,(y_g,\mathbf{h}'';);0,\mathbf{s})},$$

$$\mathbf{C}_2 = \mathbf{C}_0^{c} \cdot \mathbf{C}_1^{y_u}$$

$\mathbf{C}_2$ is calculable since $f_1^c, f_1^{cd}, f_1^{c^3}$ and $f_1^{c^3 d}$ are given. If $T = f_1^{ac^2 d}$, the challenge ciphertext is the normal challenge ciphertext because

$$\begin{aligned}
\mathbf{C}_1 &= (f_1^{ac^2 d})^{\mathbf{c}(y,(1,\mathbf{h}';);1,0)}(f_1^{ac^2})^{\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{s})}(f_1^{ad})^{\mathbf{c}(y,(y_g,\mathbf{h}'');1,0)}(f_1^{a})^{\mathbf{c}(y,(y_g,\mathbf{h}'');0,\mathbf{s})} \\
&= f_1^{ac^2\cdot\mathbf{c}(y,(1,\mathbf{h}');d,\mathbf{0})} f_1^{a\cdot c^2\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{s})} f_1^{a\cdot\mathbf{c}(y,(y_g,\mathbf{h}'');d,\mathbf{0})} f_1^{a\cdot\mathbf{c}(y,(y_g,\mathbf{h}'');0,\mathbf{s})} && (3) \\
&= f_1^{a\cdot\mathbf{c}(y,(c^2,c^2\mathbf{h}');d,\mathbf{0})} f_1^{a\cdot\mathbf{c}(y,(c^2,c^2\mathbf{h}');0,\mathbf{s})} f_1^{a\cdot\mathbf{c}(y,(y_g,\mathbf{h}'');d,\mathbf{0})} f_1^{a\cdot\mathbf{c}(y,(y_g,\mathbf{h}'');0,\mathbf{s})} && (4) \\
&= f_1^{a\cdot\mathbf{c}(y,(c^2,c^2\mathbf{h}');d,\mathbf{s})} f_1^{a\cdot\mathbf{c}(y,(y_g,\mathbf{h}'');d,\mathbf{s})} && (5) \\
&= f_1^{a\cdot\mathbf{c}(y,(c^2+y_g),c^2\mathbf{h}'+\mathbf{h}'');d,\mathbf{s})} && (6) \\
&= g_1^{a\cdot\mathbf{c}(y,(1,\mathbf{h});d,\mathbf{s})}.
\end{aligned}$$

The equalities (4) and (6) hold due to linearity over common parameters. Also, the equalities of (3) and (5) hold due to linearity over random values.

If T is a random tuple and we let $T = f_1^{ac^2d} f_1^{\gamma}$, then $f_1^{\mathbf{c}(y,(1,\mathbf{h}');\gamma,\mathbf{0})}$ is multiplied to $\mathbf{C}_1$ above. It results

$$\mathbf{C}_1 = g_1^{a\cdot\mathbf{c}(y,(1,\mathbf{h});d,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}');\gamma,\mathbf{0})}, \quad \mathbf{C}_2 = g_1^{(c+ay_u)\cdot\mathbf{c}(y,(1,\mathbf{h});d,\mathbf{s})} f_1^{y_u\mathbf{c}(y,(1,\mathbf{h}');\gamma,\mathbf{0})}.$$

Therefore, $\mathbf{C}_1$ is either a normal ciphertext or an E_0 ciphertext depending on the value of T. Therefore, if $T = f_1^{ac^2d}$, the algorithm has properly simulated Game_{Real}. Otherwise, it has simulated $\mathsf{Game}_{Real,0}$. □

In Lemma 1.2, we will show that the invariance of $\mathsf{Game}_{Real,0}$ and Game_0. Game_0 is identical with Game_{Real,m_s} because E_{m_s} ciphertext is identical with a semi-functional ciphertext by the definitions if we write $\mathcal{R}_s = \mathbb{Z}_p^{m_s}$. Therefore, we can show the invariance of two games by showing $\mathsf{Game}_{Real,i-1}$ and $\mathsf{Game}_{Real,i}$ for all $i \in [1, m_s]$.

Lemma 1.2. Suppose there exists a PPT adversary $\mathcal{A}$ who distinguishes $\mathsf{Game}_{Real,k-1}$ and Game_k with a non-negligible advantage ϵ, then an algorithm $\mathcal{B}$ which breaks $LW1$ can be built with the advantage ϵ using $\mathcal{A}$.

Proof: Using the given instances from *LW1* assumption, $\mathcal{B}$ will simulate $\mathsf{Game}_{Real,k-1}$ and $\mathsf{Game}_{Real,k}$ depending on the value of T using $\mathcal{A}$ who breaks the assumption with non-negligible advantage.

Setup and Phase I and II are simulated identically to those of Lemma 1.1.

Challenge: When the adversary asks the challenge ciphertext with messages M_0 and M_1. $\mathcal{B}$ randomly chooses $\beta \in \{0,1\}$ and $s, s_1, ..., s_{k-1}, s_{k+1}, ...s_{m_s}, s', s'_1, ..., s'_{k-1} \in \mathbb{Z}_p$. It sets $\mathbf{s} = (s_1, ..., s_{k-1}, d, s_{k+1}, ..., s_{m_s})$ and $\mathbf{s}'_{k-1} = (s'_1, ..., s'_{k-1}, 0, ..., 0)$. The value of d has never been revealed. Therefore, $\mathbf{s}$ is uniformly random to the adversary. It calculates the challenge ciphertext as

$$C = M_\beta \cdot e(f_1^{c^3 s}, f_2^c)^\alpha e(f_1^{c^2 s}, f_2)^{2\alpha\cdot y_g} e(f_1^s, f_2)^{\alpha\cdot y_g^2},$$

$$\mathbf{C}_0 = (f_1^{c^2})^{\mathbf{c}(y,(1,\mathbf{h}');s,\mathbf{s}-d\cdot\mathbf{1}_k)} (f_1^{c^2 d})^{\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{1}_k)} f_1^{\mathbf{c}(y,(y_g,\mathbf{h}'');s,\mathbf{s}-d\cdot\mathbf{1}_k)} (f_1^d)^{\mathbf{c}(y,(y_g,\mathbf{h}'');0,\mathbf{1}_k)}$$

$$\begin{aligned}\mathbf{C}_1 =& (f_1^{ac^2})^{\mathbf{c}(y,(1,\mathbf{h}');s,\mathbf{s}-d\cdot\mathbf{1}_k)} T^{\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{1}_k)} (f_1^a)^{\mathbf{c}(y,(y_g,\mathbf{h}'');s,\mathbf{s}-d\cdot\mathbf{1}_k)} \\ & \cdot (f_1^{ad})^{\mathbf{c}(y,(y_g,\mathbf{h}'');0,\mathbf{1}_k)} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1})}\end{aligned}$$

and sets $\mathbf{C}_2 = \mathbf{C}_0^c \cdot \mathbf{C}_1^{y_u}$ where $\mathbf{1}_k$ is a vector of which only the k^{th} coordinate is 1 and all other coordinates are 0. It should be noted that $\mathbf{s} - d\cdot\mathbf{1}_k$ is equal to $(s_1, ..., s_{k-1}, 0, s_{k+1}, ..., s_{m_s})$. Hence, it does not have d as an coordinate. Therefore, $\mathbf{C}_0$ and $\mathbf{C}_1$ can be calculated by the given instances. Also, $\mathbf{C}_2$ is calculable since $f_1^c, f_1^{cd}, f_1^{c^3}$ and $f_1^{c^3 d}$ are given in the instance.

If $T = f_1^{ac^2d}$, the challenge ciphertext is the normal challenge ciphertext since

$$\begin{aligned}\mathbf{C}_1 &= f_1^{ac^2\mathbf{c}(y,(1,\mathbf{h}');s,\mathbf{s}-d\cdot\mathbf{1}_k)} f_1^{ac^2 d\mathbf{c}(y,(1,\mathbf{h}');0,\mathbf{1}_k)} f_1^{a\mathbf{c}(y,(y_g,\mathbf{h}'');s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1})} \\ &= f_1^{a\mathbf{c}(y,(c^2,c^2\mathbf{h}');s,\mathbf{s})} f_1^{a\mathbf{c}(y,(y_g,\mathbf{h}'');s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1})} \\ &= f_1^{a\mathbf{c}(y,(c^2+y_g,c^2\mathbf{h}'+\mathbf{h}'');s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1})} \\ &= g_1^{a\mathbf{c}(y,(1,\mathbf{h}),s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1})}\end{aligned}$$

Otherwise, we let $T = f_1^{ac^2d} f_1^{\gamma}$. Then, $f_1^{\mathbf{c}(y,(1,\mathbf{h}');0,\gamma\cdot\mathbf{1}_k)}$ is multiplied to $\mathbf{C}_1$. It means that

$$\begin{aligned}\mathbf{C}_1 &= g_1^{a\mathbf{c}(y,(1,\mathbf{h}),s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1})} f_1^{\mathbf{c}(y,(1,\mathbf{h}');0,\gamma\cdot\mathbf{1}_k)} \\ &= g_1^{a\mathbf{c}(y,(1,\mathbf{h}),s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_{k-1}+\gamma\mathbf{1}_k)} = g_1^{a\mathbf{c}(y,(1,\mathbf{h}),s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}'),s',\mathbf{s}'_k)}.\end{aligned}$$

where $\mathbf{s}'_k = (s'_1, ..., s'_{k-1}, \gamma, 0, ...0)$.

Therefore, if $T = f_1^{ac^2d}$, the type of the challenge ciphertext is E_{k-1} and the algorithm has properly simulated $\mathsf{Game}_{Real,k-1}$. Otherwise, the type is E_k and $\mathsf{Game}_{Real,k}$ has been simulated. □

Table 4. The summary of semi-functional key invariance

Exponents of semi-functional elements of the i^{th} key for x	Type of the i^{th} key	Games	Remarks
$\mathbf{k}(0, x, (1, \mathbf{h}'); (0, ..., 0))$	Normal (=NE_0)	Game_{i-1} (=$\mathsf{Game}^N_{i,0}$)	
$\mathbf{k}(0, x, (1, \mathbf{h}'); (\boxed{r'_1}, 0, 0, ..., 0))$	NE_1	$\mathsf{Game}^N_{i,1}$	$\mathsf{Game}^N_{i,j-1} \approx$
$\mathbf{k}(0, x, (1, \mathbf{h}'); (r'_1, \boxed{r'_2}, 0, ..., 0))$	NE_2	$\mathsf{Game}^N_{i,2}$	$\mathsf{Game}^N_{i,1}$
...	...	...	by Lemma 2.1
$\mathbf{k}(0, x, (1, \mathbf{h}'); (r'_1, r'_2, r'_3..., \boxed{r'_{m_r}}))$	NSF (=NE_{m_r})	Game^N_i (=Game^N_{i,m_r})	
$\mathbf{k}(\boxed{\alpha'}^*, x, (1, \mathbf{h}'); (r'_1, ..., r'_{m_r-2}, r'_{m_r-1}, r'_{m_r}))$	TSF (=TE_{m_r})	Game^T_i (=Game^T_{i,m_r})	
$\mathbf{k}(\alpha', x, (1, \mathbf{h}'); (r'_1, ..., r'_{m_r-2}, r'_{m_r-1}, \boxed{0}))$	TE_{m_r-1}	$\mathsf{Game}^T_{i,m_r-1}$	$\mathsf{Game}^T_{i,j} \approx$
$\mathbf{k}(\alpha', x, (1, \mathbf{h}'); (r'_1, ..., r'_{m_r-2}, \boxed{0}, 0))$	TE_{i,m_r-2}	$\mathsf{Game}^T_{i,m_r-2}$	$\mathsf{Game}^T_{i,j-1}$
...	...	...	by Lemma 2.4
$\mathbf{k}(\alpha', x, (1, \mathbf{h}'); (\boxed{0}, ..., 0, 0, 0))$	SF (=TE_0)	Game_i (=$\mathsf{Game}^T_{i,0}$)	

* : α' is projected by computational α hiding in Lemmas 2.2 and 2.3.

6.2 Semi-functional Key Invariance

To prove the invariance between Game_{i-1} and Game_i, we define the security games $\mathsf{Game}^N_{i,j}$ and $\mathsf{Game}^T_{i,j}$ for $j \in [0, m_r]$ using NE_j and TE_j keys.

$\mathsf{Game}^N_{i,j}$ This game is identical with $\mathsf{Game}^N_{i,j-1}$ except the types of the i^{th} key. In this game, the i^{th} key is an NE_j key.

$\mathsf{Game}^T_{i,j}$ This game is identical with $\mathsf{Game}^T_{i,j+1}$ except the types of the i^{th} key. In this game, the i^{th} key is a TE_j key.

It should be noted that $\mathsf{Game}^N_{i,0}$ and Game^N_{i,m_r} are identical to Game_{i-1} and Game^N_i resp. by the definitions of keys. Also, due to the same reason, $\mathsf{Game}^T_{i,0}$ is equal to Game_i and Game^T_{i,m_r} is equal to Game^T_i. To prove this invariance, the type of the i^{th} key changes as Table 4. The invariance between $\mathsf{Game}^N_{i,j-1}$ and $\mathsf{Game}^N_{i,j}$ are proved in Lemma 2.1. Also, the invariance between Game^N_i and Game^T_i key is proved by Lemmas 2.2 and 2.3 using computational α hiding. Finally, the invariance between $\mathsf{Game}^T_{i,j}$ and $\mathsf{Game}^T_{i,j-1}$ is showed in Lemma 2.4.

Lemma 2. (Semi-functional Key Invariance) Suppose there exists a PPT adversary who distinguishes Game_{k-1} and Game_k with a non-negligible advantage ϵ, then an algorithm which breaks $LW2$ or computational α hiding can be built with the advantage ϵ using the adversary.

Proof: This lemma is proved by Lemmas 2.1. to 2.4. □

Lemma 2.1. Suppose there exists a PPT adversary $\mathcal{A}$ who distinguishes $\mathsf{Game}^N_{k,j-1}$ and $\mathsf{Game}^N_{k,j}$ with a non-negligible advantage ϵ, then an algorithm $\mathcal{B}$ which breaks $LW2$ can be built with the advantage ϵ using $\mathcal{A}$.

Proof: Using the given instance $\{f_1, f_1^d, f_1^{d^2}, f_1^{tw}, f_1^{dtw}, f_1^{d^2t} \in G_1, f_2,\ f_2^c, f_2^d, f_2^w, T \in G_2\}$, $\mathcal{B}$ will simulate either $\mathsf{Game}^N_{k,j-1}$ or $\mathsf{Game}^N_{k,j}$ using $\mathcal{A}$ to break $LW2$.

Setup: $\mathcal{B}$ randomly chooses $\alpha \in \mathbb{Z}_p, a, y'_v \in \mathbb{Z}_p, \mathbf{h}', \mathbf{h}'' \in \mathcal{H}$. Then, it sets MSK as $g_2 = f_2^d, g_2^\alpha = (f_2^d)^\alpha$, $g_2^{\mathbf{h}} = (f_2^d)^{\mathbf{h}'} f_2^{\mathbf{h}''}$, $v_2 = f_2^d (f_2^w)^{-a} f_2^{y'_v}$, $u_2 = f_2^w$, f_2. This implicitly sets $y_v = d - aw + y'_v$, $y_u = w$ and $b = 1/d$. Hence, it also sets $\tau = d - aw + y'_v + aw = d + y'_v$, implicitly. It publishes public parameters as

$$g_1 = f_1^d, g_1^{\mathbf{h}} = (f_1^d)^{\mathbf{h}'} f_1^{\mathbf{h}''}, g_1^a, g_1^{a \cdot \mathbf{h}}, g_1^\tau = f_1^{d^2} (f_1^d)^{y'_v},$$

$$g_1^{\tau \cdot \mathbf{h}} = (f_1^{d^2})^{\mathbf{h}'} (f_1^d)^{\mathbf{h}'' + y'_v \mathbf{h}'} (f_1)^{y'_v \mathbf{h}''}, e(g_1, g_2)^\alpha = e(f_1^d, f_2^d)^\alpha.$$

Phase I and II: The algorithm knows all MSK. Therefore, it can create the normal keys for ($> k$). For the first $k-1$ key ($< k$), $\mathcal{B}$ first generates a normal key. Then, it randomly selects α' from $\mathbb{Z}_p$ and creates an SF key. This is possible since $\mathcal{B}$ knows a, α', x and f_2.

For the k^{th} key, it randomly selects $\mathbf{z}'$ from $\mathbb{Z}_p^{m_k}$ where $m_k = |\mathbf{k}|$ and sets $\mathbf{z} = \mathbf{z}' + \mathbf{k}(0, x, (1, \mathbf{h}'); c \cdot \mathbf{1}_j)$ where $\mathbf{1}_j$ is a vector of which only the j^{th} coordinate is 1 and all other coordinates are 0. Then, it randomly chooses $\mathbf{r}''$ from $\mathcal{R}_r$ and sets $\mathbf{r} = \mathbf{r}'' - c \cdot \mathbf{1}_j$. $\mathbf{z}$ and $\mathbf{r}$ are randomly distributed because of $\mathbf{z}'$ and $\mathbf{r}''$. It also generates $r'_1, ..., r'_{j-1}$ from $\mathbb{Z}_p$ and sets $\mathbf{r}'_{j-1} = (r'_1, ..., r'_{j-1}, 0, 0, 0) \in \mathcal{R}_r$.

$$\mathbf{K}_0 = (f_2^d)^{\mathbf{k}(\alpha, x, (1,\mathbf{h}'); \mathbf{r}'')} f_2^{\mathbf{k}(0, x, (0,\mathbf{h}''); \mathbf{r}'')} (f_2^c)^{-\mathbf{k}(0, x, (0,\mathbf{h}''); \mathbf{1}_j)} (f_2^d (f_2^w)^{-a} f_2^{y'_v})^{\mathbf{z}'} \cdot T^{-a\mathbf{k}(0, x, (1,\mathbf{h}'); \mathbf{1}_j)} (f_2^c)^{y'_v \mathbf{k}(0, x, (1,\mathbf{h}'); \cdot \mathbf{1}_j)} f_2^{-a\mathbf{k}(0, x, (1,\mathbf{h}'); \mathbf{r}'_{j-1})}$$

$$\mathbf{K}_1 = (f_2^w)^{\mathbf{z}'} T^{\mathbf{k}(0, x, (1,\mathbf{h}'); \mathbf{1}_j)} f_2^{\mathbf{k}(0, x, (1,\mathbf{h}'); \mathbf{r}'_{j-1})}, \quad \mathbf{K}_2 = f_2^{-\mathbf{z}'} (f_2^c)^{-\mathbf{k}(0, x, (1,\mathbf{h}'), \mathbf{1}_j)}$$

If $T = f_2^{cw}$, then this is a properly distributed NE_{j-1} key by linearities over common parameters and random values (see Appendix A). Otherwise, if T is a random and we let $f_2^{cw+\gamma}$ denote T, this is the properly distributed NE_j key since this implicitly sets $\mathbf{r}'_j = \mathbf{r}'_{j-1} + \gamma \cdot \mathbf{1}_j$. It is worth noting that $\mathbf{r}'_j$ is uniformly random because γ is randomly distributed.

Challenge: When the adversary requests the challenge ciphertext with two message M_0 and M_1, $\mathcal{B}$ randomly selects β from $\{0, 1\}$. Then, it randomly selects $s'', \tilde{s} \in \mathbb{Z}_p$ and $\mathbf{s}'', \tilde{\mathbf{s}} \in \mathcal{R}_s$. Then, it implicitly sets $s = wt\tilde{s} + s''$, $s' = -d^2 t\tilde{s}$, $\mathbf{s}' = wt\tilde{\mathbf{s}} + \mathbf{s}''$ and $\mathbf{s}' = -d^2 t\tilde{\mathbf{s}}$. Because of $s'', \tilde{s}$, $\tilde{\mathbf{s}}$ and $\mathbf{s}''$, they are randomly distributed. $\mathcal{B}$ sets $C = M_\beta \cdot e(f_1^{dwt}, f_2^d)^{\alpha\tilde{s}} e(f_1^d, f_2^d)^{\alpha s''}$ and the others as

$$\mathbf{C}_0 = (f_1^{dwt})^{\mathbf{c}(y, (1,\mathbf{h}'); \tilde{s}, \tilde{\mathbf{s}})} (f_1^d)^{\mathbf{c}(y, (1,\mathbf{h}'); s'', \mathbf{s}'')} (f_1^{wt})^{\mathbf{c}(y, (0,\mathbf{h}''); \tilde{s}, \tilde{\mathbf{s}})} f_1^{\mathbf{c}(y, (0,\mathbf{h}''); s'', \mathbf{s}'')},$$

$$\mathbf{C}_1 = (C_0)^a (f_1^{d^2 t})^{-\mathbf{c}(y, (1,\mathbf{h}'); \tilde{s}, \tilde{\mathbf{s}})},$$

$$\mathbf{C}_2 = (f_1^{d^2})^{\mathbf{c}(y, (1,\mathbf{h}'); s'', \mathbf{s}'')} (f_1^{dwt})^{\mathbf{c}(y, (y'_v, \mathbf{h}'' + y'_v \mathbf{h}'); \tilde{s}, \tilde{\mathbf{s}})} (f_1^d)^{c(y, (y'_v, \mathbf{h}'' + y'_v \mathbf{h}'); s'', \mathbf{s}'')} \cdot (f_1^{wt})^{\mathbf{c}(y, (0, y'_v \mathbf{h}''); \tilde{s}, \tilde{\mathbf{s}})} f_1^{\mathbf{c}(y, (0, y'_v \mathbf{h}''); s'', \mathbf{s}'')}.$$

This is the properly distributed challenge ciphertext (see Appendix A). □

Lemma 2.2. Suppose there exists a PPT adversary who distinguishes Game_k^N and Game_k^T with a non-negligible advantage ϵ for $k \leq q_1$ where q_1 is the number of key queries in Phase I, then an algorithm which breaks co-selective α hiding can be built with the advantage ϵ using the adversary.

Proof: Since $k \leq q_1$, a *type k* query is queried before the *type c* query, since the k^{th} key is requested in advance of the challenge ciphertext. Hence, $\mathcal{B}$ breaks *co-selective α hiding* (i.e. $\mathcal{B}$ distinguishes whether the oracle it works with is O^1_{CAH} and O^2_{CAH}.) using the adversary $\mathcal{A}$ who distinguishes between Game_k^N and Game_k^T.

Setup: $\mathcal{B}$ makes an initial query to the oracle it works with. The oracle replies $\{f_1, f_2\}$. Then, $\mathcal{B}$ randomly selects $\alpha \in \mathbb{Z}_p, y_g, a, b, y_u, y_v, h_1, .., h_n \in \mathbb{Z}_p$ and set $\tau = y_v + a \cdot y_u$, $\mathbf{h} = (h_1, .., h_n)$, $b = y_g^{-1}$, $g_1 = f_1^{y_g}$ and $g_2 = f_2^{y_g}$. It publishes public parameters $\{g_1, g_1^{\mathbf{h}}, g_1^a, g_1^{a \cdot \mathbf{h}}, g_1^\tau, g_1^{\tau \cdot \mathbf{h}}, e(g_1, g_2)^\alpha\}$. It sets MSK as $\{g_2, g_2^\alpha, g_2^{\mathbf{h}}, v_2 = f_2^{y_v}, u_2 = f_2^{y_u}, f_2\}$.

Phase I and II: For the first k-1 keys, $\mathcal{B}$ generates a normal key $(\mathbf{K}'_0, \mathbf{K}'_1, \mathbf{K}'_2)$ using the key generation algorithm and MSK. It randomly chooses α' from $\mathbb{Z}_p$ and sets

$$\mathbf{K}_0 = \mathbf{K}'_0 \cdot f_2^{-a \cdot \mathbf{k}(\alpha', x, (0,\mathbf{0}); \mathbf{0})}, \mathbf{K}_1 = \mathbf{K}'_1 \cdot f_2^{\mathbf{k}(\alpha', x, (0,\mathbf{0}); \mathbf{0})}, \mathbf{K}_2 = \mathbf{K}'_2.$$

For the rest of keys except the k^{th} key ($> k$), the algorithm responds to the key queries by sending a normal key. This is possible since $\mathcal{B}$ knows all PK and MSK.

When the adversary requests the k^{th} key, the algorithm creates a normal key $(\mathbf{K}'_0, \mathbf{K}'_1, \mathbf{K}'_2)$ and requests a type k response to the oracle it works with. We let $f_2^{\mathbf{k}(\alpha'', x, (1,\mathbf{h}'); \mathbf{r}')}$ denote the response

from the oracle. $\mathcal{B}$ does not know whether α'' is 0 or a random value since it does not know the type of the oracle. It sets

$$\mathbf{K}_0 = \mathbf{K}'_0(f_2^{\mathbf{k}(\alpha'',x,(1,\mathbf{h}');\mathbf{r}')})^{-a}, \quad \mathbf{K}_1 = \mathbf{K}'_1 f_2^{\mathbf{k}(\alpha'',x,(1,\mathbf{h}');\mathbf{r}')}, \quad \mathbf{K}_2 = \mathbf{K}'_2.$$

Finally, it sends the key to the adversary. It is worth noting that k^{th} key is queried only in phase I since $k \leq q_1$. Therefore, if $\alpha'' = 0$, the oracle which $\mathcal{B}$ works with is O^1_{CAH}. If α'' is a random, it works with O^2_{CAH}.

Challenge: When the adversary requests the challenge ciphertext with two message M_0 and M_1, the algorithm randomly selects β from $\{0,1\}$, the algorithm first creates a normal ciphertext $(C', \mathbf{C}'_0, \mathbf{C}'_1, \mathbf{C}'_2)$. Then, it requests a *type c* response to the oracle it works with. We let $f_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')}$ denote the response from the oracle. It sends to the adversary

$$C = C', \mathbf{C}_0 = \mathbf{C}'_0, \mathbf{C}_1 = \mathbf{C}'_1 f_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')}, \mathbf{C}_2 = \mathbf{C}'_2 (f_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')})^{y_u}.$$

□

Lemma 2.3. Suppose there exists a PPT adversary who distinguishes Game^N_k and Game^T_k with a non-negligible advantage ϵ for $k > q_1$ where q_1 is the number of key queries in Phase I, then an algorithm which breaks selective α hiding can be built with the advantage ϵ using the adversary.

Proof: This proof is almost identical with Lemma 2.2 except the type of the oracle the algorithm works with and the k^{th} key is simulated in phase II ($k > q_1$). In this proof, $\mathcal{B}$ works with either O^1_{SAH} or O^2_{SAH} depending on the value of α'' since it queries a type c in advance of a type k. □

Lemma 2.4. Suppose there exists a PPT adversary who distinguishes $\mathsf{Game}^T_{k,j}$ and $\mathsf{Game}^T_{k,j-1}$ with a non-negligible advantage ϵ, then an algorithm which breaks $LW2$ can be built with the advantage ϵ using the adversary.

Proof: This proof is identical with the proof of Lemma 2.1 except the k^{th} key. For the k^{th} key, it randomly selects $\alpha' \in \mathbb{Z}_p$ and $\mathbf{z}' \in \mathbb{Z}_p^{m_k}$ and sets $\mathbf{z} = \mathbf{z}' + \mathbf{k}(\alpha', x, (1,\mathbf{h}'), c \cdot \mathbf{1}_j)$. Then, it chooses $\mathbf{r}''$ from $\mathcal{R}_r$ and sets $\mathbf{r} = \mathbf{r}'' - c \cdot \mathbf{1}_j$. $\mathbf{z}$ and $\mathbf{r}$ are randomly distributed because of $\mathbf{z}'$ and $\mathbf{r}''$. It also generates $r'_1, ..., r'_{j-1}$ from $\mathbb{Z}_p$ and sets $\mathbf{r}'_{j-1} = (r'_1, ..., r'_{j-1}, 0, 0, 0) \in \mathcal{R}_r$.

$$\begin{aligned}\mathbf{K}_0 =& (f_2^d)^{\mathbf{k}(\alpha+\alpha',x,(1,\mathbf{h}');\mathbf{r}'')} f_2^{\mathbf{k}(0,x,(0,\mathbf{h}'');\mathbf{r})} (f_2^c)^{-\mathbf{k}(0,x,(0,\mathbf{h}'');\cdot\mathbf{1}_j)} (f_2^d (f_2^w)^{-a} f_2^{y'_v})^{\mathbf{z}'} \\ & \cdot T^{-a\mathbf{k}(\alpha',x,(1,\mathbf{h}');\cdot\mathbf{1}_j)} (f_2^c)^{y'_v \mathbf{k}(\alpha',x,(1,\mathbf{h}');\cdot\mathbf{1}_j)} f_2^{-a\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{r}'_{j-1})}\end{aligned}$$

$$\mathbf{K}_1 = (f_2^w)^{\mathbf{z}'} T^{\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{1}_j)} f_2^{\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{r}'_{j-1})}, \quad \mathbf{K}_2 = f_2^{-\mathbf{z}'} (f_2^c)^{-\mathbf{k}(\alpha',x,(1,\mathbf{h}'),\mathbf{1}_j)}$$

If $T = f_2^{cw}$, this is the properly distributed TE_{j-1} key. Therefore, $\mathcal{B}$ has well simulated $\mathsf{Game}^T_{k,j-1}$. If T is random and we let $f_2^{cw+\gamma}$ denote T, This is the properly distributed TE_j key since this implicitly sets $\mathbf{r}'_j = \mathbf{r}'_{j-1} + \gamma \cdot \mathbf{1}_j$. Due to γ, $\mathbf{r}'_j$ is properly distributed. Hence, $\mathcal{B}$ has simulated $\mathsf{Game}^T_{k,j}$. □

6.3 Semi-functional Security

We prove the semi-functional security by showing that Game_{q_t} and Game_{Final} are indistinguishable.

Lemma 3. Suppose there exists a PPT adversary who distinguishes Game_{q_t} and Game_{Final} with a non-negligible advantage ϵ, then an algorithm which breaks $LW3$ can be built with the advantage ϵ using the adversary.

Proof: Using a given instance $\{f_1, f_1^a, f_1^c, f_1^d \in G_1, f_2, f_2^a, f_2^c, f_2^d \in G_2, T \in G_T\}$, $\mathcal{B}$ will simulate either Game_{q_t} or Game_{Final} depending on the value of T.

Setup: $\mathcal{B}$ randomly selects $y_g, y_u, y_v \in \mathbb{Z}_p, \mathbf{h} \in \mathbb{Z}_p^n$ and sets $\alpha = ac$, $a = a$, $b = 1/y_g$ and $\tau = y_v + a y_u$. $\mathcal{B}$ publishes the public parameters

$$g_1 = f_1^{y_g}, g_1^{\mathbf{h}} = {f_1}^{y_g \mathbf{h}}, g_1^a = f_1^{a y_g}, g_1^{a\mathbf{h}} = f_1^{a y_g \mathbf{h}},$$

$$g_1^{\tau} = f_1^{y_g y_v} + (f_1^a)^{y_g y_u}, g_1^{\tau \mathbf{h}} = f_1^{y_g y_v \mathbf{h}} + (f_1^a)^{y_g y_u \mathbf{h}}, e(g_1, g_2)^{\alpha} = e(f_1^a, f_2^c)^{y_g^2}$$

It also sets $g_2^{\alpha} = f_2^{acy_g}, g_2 = f_2^{y_g}, g_2^{\mathbf{h}} = f_2{}^{y_g \mathbf{h}}, v_2 = f_2^{y_v}, u_2 = f_2^{y_u}, f_2$. It should be noted that g_2^{α} sets implicitly since f_2^{ac} is not given. The other elements can be calculated using f_2 and f_2^a given in the instance.

Phase I and II: $\mathcal{B}$ randomly selects $\alpha'' \in \mathbb{Z}_p$, $\mathbf{z} \in \mathbb{Z}_p^{m_k}$ and $\mathbf{r} \in \mathcal{R}_s$ and sets $\alpha' = y_g c + \alpha''$.

$$\mathbf{K}_0 = f_2^{\mathbf{k}(-a\alpha'', x, (y_g, y_g \mathbf{h}); \mathbf{r})} v_2^{\mathbf{z}},$$

$$\mathbf{K}_1 = f_2^{y_u \mathbf{z}} (f_2^c)^{\mathbf{k}(y_g, x, (\mathbf{0}, \mathbf{0}), \mathbf{0})} f_2^{\mathbf{k}(\alpha'', x, (\mathbf{0}, \mathbf{0}), \mathbf{0})}, \quad \mathbf{K}_2 = f_2^{-\mathbf{z}}$$

This is a properly distributed key since

$$\begin{aligned}
\mathbf{K}_0 &= f_2^{\mathbf{k}(-a\alpha'', x, (y_g, y_g \mathbf{h}); \mathbf{r})} v_2^{\mathbf{z}} \\
&= f_2^{\mathbf{k}(y_g ac, x, (y_g, y_g \mathbf{h}); \mathbf{r})} \cdot v_2^{\mathbf{z}} \cdot f_2^{\mathbf{k}(-y_g ac - a\alpha'', x, (y_g, y_g \mathbf{h}); \mathbf{0})} && (7) \\
&= g_2^{\mathbf{k}(\alpha, x, (1, \mathbf{h}); \mathbf{r})} \cdot v_2^{\mathbf{z}} \cdot f_2^{-a\mathbf{k}(y_g c + \alpha'', x, (0,0); \mathbf{0})} && (8)
\end{aligned}$$

The equality of (7) holds due to linearity over random values. Also, (8) holds by parameter vanishing and linearity over common parameters.

Challenge: When the adversary requests the challenge ciphertexts with two messages M_0 and M_1. $\mathcal{B}$ randomly selects $\beta \in \{0,1\}$, $s'' \in \mathbb{Z}_p$, $\mathbf{s}, \mathbf{s}'' \in \mathcal{R}_s$ and $\mathbf{h}'' \in \mathbb{Z}_p^n$ and sets $s = d$, $s' = -y_g ad + as''$ and $\mathbf{s}' = a \cdot \mathbf{s}''$. d appears both in s and s'. However, s' does not reveal the value of d because of s''. Therefore, setting $s = d$ is hidden to the adversary. It calculates the challenge ciphertexts as follows:

$$C = M \cdot T, \quad \mathbf{C}_0 = f_1^{y_g \mathbf{c}(y, (1, \mathbf{h}); s, \mathbf{s})}$$

$$\mathbf{C}_1 = (f_1^a)^{\mathbf{c}(y, (1, \mathbf{h}); s'', y_g \mathbf{s} + \mathbf{s}'')} f_1^{\mathbf{c}(y, (0, \mathbf{h}''); s'', \mathbf{s}'')} (f_1^d)^{\mathbf{c}(y, (0, \mathbf{h}''); -y_g, \mathbf{0})}, \mathbf{C}_2 = \mathbf{C}_0^{y_v} \mathbf{C}_1^{y_u}$$

This implicitly sets $\mathbf{h}' = \mathbf{h} + a^{-1}\mathbf{h}''$. $\mathbf{C}_1$ and $\mathbf{C}_2$ are properly distributed since

$$\begin{aligned}
\mathbf{C}_1 &= (f_1^a)^{\mathbf{c}(y, (1, \mathbf{h}); s'', y_g \mathbf{s} + \mathbf{s}'')} f_1^{\mathbf{c}(y, (0, \mathbf{h}''); s'', \mathbf{s}'')} (f_1^d)^{\mathbf{c}(y, (0, \mathbf{h}''); -y_g, \mathbf{0})} \\
&= (f_1^a)^{\mathbf{c}(y, (1, \mathbf{h}); y_g d - y_g d + s'', y_g \mathbf{s} + \mathbf{s}'')} f_1^{\mathbf{c}(y, (0, \mathbf{h}''); -y_g d + s'', \mathbf{s}'')} && (9) \\
&= (f_1^a)^{\mathbf{c}(y, (1, \mathbf{h}); y_g d, y_g \mathbf{s})} f_1^{\mathbf{c}(y, (1, \mathbf{h}); -y_g ad + as'', a\mathbf{s}'')} f_1^{\mathbf{c}(y, (0, \mathbf{h}''/a); -y_g ad + as'', a\mathbf{s}'')} && (10) \\
&= (g_1^a)^{\mathbf{c}(y, (1, \mathbf{h}); d, \mathbf{s})} (f_1)^{\mathbf{c}(y, (1, \mathbf{h} + \mathbf{h}''/a); -y_g ad + as'', a\mathbf{s}'')} && (11)
\end{aligned}$$

The equalities of (9) and (10) holds by linearity over random values. The equality of (11) holds because of linearity over common parameters. It should be noted that $\mathbf{h}''$ does not appear anywhere else, it only used in the challenge ciphertext. Therefore, due to $\mathbf{h}''$, $\mathbf{h}'$ is randomly distributed. If T is $e(f_1, f_2)^{acd}$, this has simulated Game_{q_t} properly. Otherwise, if T is a random, a randomness will be added to M. Therefore, this has simulated Game_{Final}. □

7 Conclusion

In this paper, we provide a new construction for the pair encoding framework in prime order groups. In particular, our construction is also applicable when pair encodings require computational hiding to prove their security. Prior to our works, pair encoding framework supports a number of adaptively secure functional encryptions only in composite order groups if their property requires computational hiding, but our construction shows that those schemes can be also realised in prime order groups to improve their efficiency significantly. More specifically, our construction requires only three group elements in prime order groups to feature one group element of prior construction in composite order groups. Therefore, our works improve the efficiency of those functional encryptions significantly while they still remain to be adaptively secure.

References

1. S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. *IACR Cryptology ePrint Archive*, 2015:413, 2015.
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 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.
5. D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.
6. D. Boneh and M. K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.
7. D. Boneh, C. Gentry, S. Gorbunov, S. Halevi, V. Nikolaenko, G. Segev, V. Vaikuntanathan, and D. Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 533–556. Springer, 2014.
8. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.
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, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *LNCS*, pages 435–460. Springer, 2013.
11. C. Cocks. An identity based encryption scheme based on quadratic residues. In B. Honary, editor, *IMA Int. Conf.*, volume 2260 of *LNCS*, pages 360–363. Springer, 2001.
12. 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.
13. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.
14. G. Herold, J. Hesse, D. Hofheinz, C. Ràfols, and A. Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. In *CRYPTO*.
15. Y. Ishai and H. Wee. Partial garbling schemes and their applications. In *ICALP*.
16. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
17. 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.
18. 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.
19. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In D. Micciancio, editor, *TCC*, volume 5978 of *LNCS*, pages 455–479. Springer, 2010.
20. 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.
21. 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.
22. 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.

23. 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.
24. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
25. 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.
26. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
27. 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.
28. B. Waters. Functional encryption for regular languages. In R. Safavi-Naini and R. Canetti, editors, *CRYPTO*, volume 7417 of *LNCS*, pages 218–235. Springer, 2012.
29. H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC*, volume 8349 of *LNCS*, pages 616–637. Springer, 2014.

A Equations in Lemma 2.1.

If $T = f_2^{cw}$, then $\mathbf{K}_0, \mathbf{K}_1$ and $\mathbf{K}_2$ are properly distributed NE_{j-1} since

$$\begin{aligned}
\mathbf{K}_0 &= (f_2^d)^{\mathbf{k}(\alpha,x,(1,\mathbf{h}');\mathbf{r}'')} f_2^{\mathbf{k}(0,x,(0,\mathbf{h}'');\mathbf{r}'')} (f_2^c)^{-\mathbf{k}(0,x,(0,\mathbf{h}'');\mathbf{1}_j)} (f_2^d (f_2^{wa})^{-1} f_2^{y'_v})^{\mathbf{z}'} \\
&\quad (f^{cw})^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\cdot\mathbf{1}_j)} (f_2^c)^{y'_v \mathbf{k}(0,x,(1,\mathbf{h}');\cdot\mathbf{1}_j)} f_2^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \\
&= f_2^{d\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{r}'')} \boxed{f_2^{d\mathbf{k}(0,x,(1,\mathbf{h}');-c\mathbf{1}_j)}} f_2^{\mathbf{k}(0,x,(0,\mathbf{h}'');\mathbf{r}'')} f_2^{\mathbf{k}(0,x,(0,\mathbf{h}'');-c\mathbf{1}_j)} f_2^{(d-wa+y'_v)(\mathbf{z}')} \\
&\quad \cdot \boxed{f_2^{d\mathbf{k}(0,x,(1,\mathbf{h}');c\mathbf{1}_j)}} f_2^{-wa\mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j)} f_2^{y'_v \mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j)} f_2^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \qquad (12) \\
&= f_2^{d\mathbf{k}(\alpha',x,(1,\mathbf{h}');\mathbf{r})} f_2^{\mathbf{k}(0,x,(0,\mathbf{h}'');\mathbf{r})} f_2^{(d-wa+y'_v)(\mathbf{z}'+\mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j))} f_2^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \qquad (13) \\
&= f_2^{\mathbf{k}(d\alpha',x,(d,d\mathbf{h}');\mathbf{r})} f_2^{\mathbf{k}(0,x,(0,\mathbf{h}'');\mathbf{r})} f_2^{(d-wa+y'_v)(\mathbf{z}'+\mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j))} f_2^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \qquad (14) \\
&= f_2^{\mathbf{k}(d\alpha',x,(d,d\mathbf{h}'+\mathbf{h}'');\mathbf{r})} f_2^{(d-wa+y'_v)(\mathbf{z}'+\mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j))} f_2^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \qquad (15) \\
&= g_2^{\mathbf{k}(\alpha',x,(1,\mathbf{h});\mathbf{r})} v_2^{\mathbf{z}} f_2^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})}
\end{aligned}$$

This implicitly sets $\mathbf{r} = \mathbf{r}'' - c \cdot \mathbf{1}_j$ and $\mathbf{z} = \mathbf{z}' + \mathbf{k}(0, x, (1, \mathbf{h}'); c \cdot \mathbf{1}_j)$. The second equality (12) in above equation holds by the linearity over random values because

$$\begin{aligned}
(f_2^d)^{\mathbf{k}(\alpha,x,(1,\mathbf{h}');\mathbf{r}'')} &= (f_2^d)^{\mathbf{k}(\alpha,x,(1,\mathbf{h}');\mathbf{r}'')} (f_2^d)^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{0})} \\
&= (f_2^d)^{\mathbf{k}(\alpha,x,(1,\mathbf{h}');\mathbf{r}'')} (f_2^d)^{\mathbf{k}(0,x,(1,\mathbf{h}');-c\mathbf{1}_j)} (f_2^d)^{\mathbf{k}(0,x,(1,\mathbf{h}');c\mathbf{1}_j)}.
\end{aligned}$$

The third equality (13) holds because of the definition of $\mathbf{r}$ $(= \mathbf{r}'' - c \cdot \mathbf{1}_j)$ and linearity over random values. The equalities (14) and (15) hold due to linearity over common parameters.

$$\begin{aligned}
\mathbf{K}_1 &= (f_2^w)^{\mathbf{z}'} (f_2^{cw})^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{1}_j)} f_2^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \\
&= (f_2^w)^{\mathbf{z}'} (f_2^w)^{\mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j)} f_2^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} \\
&= (f_2^w)^{\mathbf{z}'+\mathbf{k}(0,x,(1,\mathbf{h}');c\cdot\mathbf{1}_j)} f_2^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})} = u_2^{\mathbf{z}} f_2^{\mathbf{k}(0,x,(1,\mathbf{h}');\mathbf{r}'_{j-1})}
\end{aligned}$$

If T is random in Lemma 2.1. and we let $f_2^{wc+\gamma}$ denote it, This is properly distributed NE_j since $(f_2^{\gamma})^{-a\mathbf{k}(0,x,(1,\mathbf{h}');\cdot\mathbf{1}_j)}$ is multiplied to $\mathbf{K}_1$. By linearity over random values, this implicitly sets $\mathbf{r}'_j = \mathbf{r}'_{j-1} + \gamma \cdot \mathbf{1}_j$. $\mathbf{r}'_j$ is still randomly distributed since γ is a random value.

The challenge ciphertext is also properly distributed because

$$\begin{aligned}
\mathbf{C}_0 &= (f_1^{dwt})^{\mathbf{c}(y,(1,\mathbf{h}');\tilde{s},\tilde{\mathbf{s}})} (f_1^d)^{\mathbf{c}(y,(1,\mathbf{h}');s'',\mathbf{s}'')} (f_1^{wt})^{\mathbf{c}(y,(0,\mathbf{h}'');\tilde{s},\tilde{\mathbf{s}})} f_1^{\mathbf{c}(y,(0,\mathbf{h}'');s'',\mathbf{s}'')} \\
&= (f_1^{wt})^{\mathbf{c}(y,(d,d\mathbf{h}');\tilde{s},\tilde{\mathbf{s}})} f_1^{\mathbf{c}(y,(d,d\mathbf{h}');s'',\mathbf{s}'')} (f_1^{wt})^{\mathbf{c}(y,(0,\mathbf{h}'');\tilde{s},\tilde{\mathbf{s}})} f_1^{\mathbf{c}(y,(0,\mathbf{h}'');s'',\mathbf{s}'')} \qquad (16) \\
&= (f_1^{wt})^{\mathbf{c}(y,(d,d\mathbf{h}');\tilde{s},\tilde{\mathbf{s}})} (f_1)^{\mathbf{c}(y,(d,d\mathbf{h}');s'',\mathbf{s}'')} (f_1^{wt})^{\mathbf{c}(y,(0,\mathbf{h}'');\tilde{s},\tilde{\mathbf{s}})} f_1^{\mathbf{c}(y,(0,\mathbf{h}'');s'',\mathbf{s}'')} \qquad (17) \\
&= f_1^{\mathbf{c}(y,(d,d\mathbf{h}');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')} f_1^{\mathbf{c}(y,(0,\mathbf{h}'');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')} \qquad (18) \\
&= f_1^{\mathbf{c}(y,(d,d\mathbf{h}'+\mathbf{h}'');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')} \qquad (19) \\
&= g_1^{\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})}
\end{aligned}$$

$$\mathbf{C}_1 = (C_0)^a (f_1^{d^2 t})^{-\mathbf{c}(y,(1,\mathbf{h}');\tilde{s},\tilde{\mathbf{s}})} = g_1^{a\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})} f_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')}$$

$$
\begin{aligned}
\mathbf{C}_2 &= (f_1^{d^2})^{\mathbf{c}(y,(1,\mathbf{h}');s'',\mathbf{s}'')}(f_1^{dwt})^{\mathbf{c}(y,(y'_v,\mathbf{h}''+y'_v\mathbf{h}');\tilde{s},\tilde{\mathbf{s}})}(f_1^{d})^{c(y,(y'_v,\mathbf{h}''+y'_v\mathbf{h}');s'',\mathbf{s}'')} \\
&\quad \cdot (f_1^{wt})^{\mathbf{c}(y,(0,y'_v\mathbf{h}'');\tilde{s},\tilde{\mathbf{s}})} f_1^{\mathbf{c}(y,(0,y'_v\mathbf{h}'');s'',\mathbf{s}'')} \\
&= (f_1^{d^2})^{\mathbf{c}(y,(1,\mathbf{h}');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')}(f_1^{d^2})^{\mathbf{c}(y,(1,\mathbf{h}');-wt\tilde{s},-wt\tilde{\mathbf{s}})} \\
&\quad \cdot (f_1^{d})^{\mathbf{c}(y,(y'_v,\mathbf{h}''+y'_v\mathbf{h}');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')} f_1^{\mathbf{c}(y,(0,y'_v\mathbf{h}'');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')} \qquad (20) \\
&= f_1^{\mathbf{c}(y,((d+y'_v)d,d(d+y'_v)\mathbf{h}'+(d+y'_v)\mathbf{h}'');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')}(f_1^{w})^{\mathbf{c}(y,(1,\mathbf{h}');-d^2t\tilde{s},-d^2t\tilde{\mathbf{s}})} \qquad (21) \\
&= g_1^{\tau\mathbf{c}(y,(1,\mathbf{h});s,\mathbf{s})} u_1^{\mathbf{c}(y,(1,\mathbf{h}');s',\mathbf{s}')}.
\end{aligned}
$$

The equalities of (16) and (19) hold by linearity over common parameters. Also, those of (17) and (18) hold by linearity over random values. The equalities of (20) holds since

$$
\begin{gathered}
(f_1^{d^2})^{\mathbf{c}(y,(1,\mathbf{h}');s'',\mathbf{s}'')} = (f_1^{d^2})^{\mathbf{c}(y,(1,\mathbf{h}');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')}(f_1^{d^2})^{\mathbf{c}(y,(1,\mathbf{h}');-wt\tilde{s},-wt\tilde{\mathbf{s}})} \\
(f_1^{dwt})^{\mathbf{c}(y,(y'_v,\mathbf{h}''+y'_v\mathbf{h}');\tilde{s},\tilde{\mathbf{s}})}(f_1^{d})^{c(y,(y'_v,\mathbf{h}''+y'_v\mathbf{h}');s'',\mathbf{s}'')} = (f_1^{d})^{\mathbf{c}(y,(y'_v,\mathbf{h}''+y'_v\mathbf{h}');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')} \\
(f_1^{wt})^{\mathbf{c}(y,(0,y'_v\mathbf{h}'');\tilde{s},\tilde{\mathbf{s}})} f_1^{\mathbf{c}(y,(0,y'_v\mathbf{h}'');s'',\mathbf{s}'')} = f_1^{\mathbf{c}(y,(0,y'_v\mathbf{h}'');wt\tilde{s}+s'',wt\tilde{\mathbf{s}}+\mathbf{s}'')}.
\end{gathered}
$$

It is worth noting that all equalities above hold by linearity over random values. The last equalities in $\mathbf{C}_0$, $\mathbf{C}_1$ and $\mathbf{C}_2$ hold because of $s' = -d^2t\tilde{s}$, $\mathbf{s}' = -d^2t\tilde{\mathbf{s}}$ and the definitions of public parameters. $\tilde{s}$ and $\tilde{\mathbf{s}}$ are randomly distributed to the adversary although they also appear in $s = wt\tilde{s} + s'', \mathbf{s} = wt\tilde{s} + \mathbf{s}''$ since their values are not revealed in those values (due to s'' and $\mathbf{s}''$).