A New Encoding Framework for Predicate Encryption with Non-linear Structures in Prime Order Groups

Jongkil Kim[1(✉)], Willy Susilo[1], Fuchun Guo[1], Joonsang Baek[1], and Nan Li[2]

[1] Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, Australia
{jongkil,wsusilo,fuchun,baek}@uow.edu.au

[2] School of Electrical Engineering and Computing, The University of Newcastle, Newcastle, Australia
nan.li@newcastle.edu.au

Abstract. We present a new encoding framework for predicate encryption (PE) in prime order groups. Our framework captures a broader range of adaptively secure PE schemes by allowing PE schemes to have more flexible (i.e., non-linear) structures. The existing works dealing with adaptively secure PE schemes in prime order groups require strict structural restrictions on PE schemes. In particular, the exponents of public keys and master secret keys of the PE schemes, which are referred to as common variables, must be linear. In this paper, we introduce a modular approach which includes non-linear common variables in PE schemes. First, we formalize non-linear structures by improving Attrapadung's pair encoding framework (Eurocrypt'14). Then, we provide a generic compiler that incorporates encodings under our framework to PE schemes in prime order groups. Notably, we prove the security of our compiler by introducing a new technique that decomposes common variables into two types and makes one of them shared between semi-functional and normal spaces on processes of the dual system encryption. As instances of our new framework, we introduce new attribute-based encryption schemes supporting non-monotone access structures, namely non-monotonic ABE. Our new schemes are adaptively secure in prime order groups and have either short ciphertexts (in the case of KP-ABE) or short keys (in the case of CP-ABE).

Keywords: Pair encoding · Non-monotone access structure · Attribute-based encryption · Prime order groups · Dual system encryption

1 Introduction

Wee [18] and Attrapadung [3] introduced generic modular frameworks which generalize predicate encryption (PE) using encodings. They extracted common properties that PE schemes shared and formalized them under the encoding frameworks. Their encoding frameworks include generic constructions (i.e., compilers) of PE schemes based on encodings and approaches to proofs of adaptive security only using the properties the encodings commonly have. Therefore, these frameworks give a new insight into building PE schemes as the security of PE schemes can be proven by showing that their corresponding encoding schemes satisfy those properties.

Recently, encoding frameworks have been adopted to find a generic construction in prime order groups [1,2,5,11,14]. The benefit of the prime order groups is the efficiency gains that they can bring to encryption schemes. However, the constructions based on the prime order groups commonly impose a more structural restriction on encoding schemes. In particular, they require the exponents of public and master secret keys (which are referred to as common variables) to have a simple linear structure.

For example, if we denote the common variables of an encoding scheme by $h_1, ..., h_m$, the constructions require that public and master secret keys to be set as $g, g^{h_1}, ..., g^{h_m}$ where g is a group generator. Note that they cannot allow encoding schemes to have the parameters of group elements whose exponents are not linear in h_i such as $g^{h_1^2}$ or $g^{h_1 h_2}$. This is because most of the known techniques in prime order groups require parameters in an encryption scheme to be represented using matrices. Hence, the multiplication between parameters cannot be easily handled since those matrices do not commute. It adds more restrictions on the structures of the encoding scheme and limits the usage of encoding frameworks.

1.1 Our Contribution

Framework with Less Structural Requirement. We introduce a modular framework which is applicable to PE schemes having non-linear common variables in prime order groups. Prior to our work, existing frameworks [1,2,5,11,14] in prime order groups covers PE schemes which have a simple linear structure. Our new framework overcomes this barrier by suggesting a new framework and a new proof technique. To mitigate the structural restriction and effectively express non-linearity of PE schemes, we improve Attrapadung's pair encoding framework [3] which is one of the most popular encoding frameworks for PE and provide a new adaptively secure compiler that incorporates an encoding scheme under our improved framework to a PE scheme in prime order groups.

ABEs with a Non-monotone Access Structure. As instances of our new encoding technique, we introduce two new attribute-based encryption (ABE) schemes supporting a non-monotone access structure as follows:

- Non-monotonic CP-ABE (NM-CP-ABE) with short keys (Scheme 1).
- Non-monotonic KP-ABE (NM-KP-ABE) with short ciphertexts (Scheme 2).

Note that although Yamada et al. already introduced selectively secure schemes in [26], no encoding framework was able to achieve adaptive security in prime order groups due to the non-linearity. For the first time, our new schemes achieve non-monotone access structure, short parameters (key or ciphertexts) and adaptive security at the same time. Table 1 summarizes comparison between our schemes and the existing non-monotonic ABE schemes.

Table 1. Comparisons of Non-monotonic ABE schemes in prime order groups

Scheme	Multi-use of Att.	Security	Assumptions	Type	NM-CP-ABE	
					CT	Priv. Key
LSW [16]	Yes	Selective	RO+n-MEBDH	KP	$3n+1$	$2t+t'$
AHLLPR [6]	Yes	Selective	n-DBDHE	KP	4	$(N+1)t$
YAHK [26]	Yes	Selective	q-types	CP	$3t+1$	$4n+2$
	Yes	Selective	q-types	KP	$4n+1$	$3t$
OT [23]	No	Adaptive	DLIN	CP	$14t+5$	$14n\tilde{u}+5$
	No	Adaptive	DLIN	KP	$14n\tilde{u}+5$	$14t+5$
Scheme 1	Yes	Adaptive	Static + q-types	CP	$3(N+2)t+6$	21
Scheme 2	Yes	Adaptive	Static + q-types	KP	24	$3(N+2)t+9$

t: the number of attributes in an access policy, t': the number of negated attributes in an access policy,
n: the number of attributes in attribute sets, N: the maximum number of attributes in attribute sets
$\tilde{u}$: the maximum number of appearances of an attribute in an access policy.
Static: 'Static' in Assumptions implies that $LW1$, $LW2$ and DBDH

1.2 Overview of Our Technique

Main Idea. Our solution largely adopts the notion of pair encoding framework, which is outlined in Appendix A.1. However, the pair encoding framework cannot properly describe non-linear common variables. Therefore, we modify the syntax of pair encoding to be more flexible. The most significant change in our framework is decomposing common variables in the pair encoding framework into *hidden common variables* and *shared common variables* as we describe below:

- *Hidden Common Variables (HCVs)* are identical to common variables used in existing frameworks [1,5,11,14]. The HCVs must be linear.
- *Shared Common Variables (SCVs)* are variables which are non-linear or cause a non-linearity.

In detail, the exponents of public parameters and master secrets in our encoding framework are the composition of those two types of common variables. We use $\boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}) = (b_1, ..., b_\omega)$ to denote the exponents of those parameters and also use $\boldsymbol{w} = (w_1, ..., w_{\omega_1})$ and $\boldsymbol{h} = (h_1, ..., h_{\omega_2})$ to denote SCVs and HCVs, respectively. b_i is defined as a monomial which is $b_i = b_0 f_i(\boldsymbol{w})$ or $f_i(\boldsymbol{w})h_j$ where $f_i(\boldsymbol{w})$ is a monomial consisting of the elements of $\boldsymbol{w}$ and $j \in [\omega_2]$ and b_0 is a variable adopted for a linear operation of monomials where HCVs do not appear. This setting makes $\boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h})$ linear in $(b_0, \boldsymbol{h})$. More formally, by the definition of $\boldsymbol{b}$, for all $b_0, b_0' \in \mathbb{Z}_p$ and $\boldsymbol{h}, \boldsymbol{h}' \in \mathbb{Z}_p^{\omega_2}$, we have

$$\boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}) + \boldsymbol{b}(\boldsymbol{w}, b_0', \boldsymbol{h}') = \boldsymbol{b}(\boldsymbol{w}, b_0 + b_0', \boldsymbol{h} + \boldsymbol{h}').$$

We call this property *linearity in HCVs.*

HCVs and SCVs work differently in the security proof. Encoding frameworks can be considered as generalizations of Waters' dual system encryption [25]. In the dual system encryption, *semi-functional space* is used to partially mimic the construction of an encryption scheme to prove the security more simply, but variables appeared in semi-functional space must not correlate with their original values in the construction, which we call normal space. HCVs are variables which are typically used in the dual system encryption. They are projected from normal space to semi-functional space in the proofs. Their values in semi functional space do not correlate to their original values. However, SCVs are a new type of variables. They are also projected to the semi-functional space, but their projected values are identical to their original values. This is possible since the proof works in a prime order group. In other words, SCVs are shared both in semi-functional and normal spaces, where the construction is defined. We handle these changes by refining the security proof and the property of encodings.

Parameter b_0. Additionally, due to the notational deficiency of the pair encoding to express the linearity of (hidden) common variables, we have adopted a new variable b_0 in our encoding framework as done in Kim et al.'s work [14]. Speaking more precisely, even if HCVs of $\boldsymbol{b}$ are linear form (i.e. the maximum degree of those variables is set to be 1), the linearity in HCVs of $\boldsymbol{b}$ cannot properly be notated if coordinates of $\boldsymbol{b}$ do not have an element of $\boldsymbol{h}$. Thus, we use a new variable b_0 to denote the change the values during the linear operation and place b_0 where an element of $\boldsymbol{h}$ does not appear. Consequently, all coordinates of $\boldsymbol{b}$ must contain either b_0 or h_i and linear in those variables.

Our Compiler in Prime Order Groups. To construct a new compiler of encodings with a less restrictive structural assumption, we adopt the technique from [14], in which the common variables are projected into semi-functional space. This technique is built upon combining a nested dual system encryption technique and Lewko and Waters' IBE [17]. In particular, the simulator sets a common variable as $d \cdot \boldsymbol{h}' + \boldsymbol{h}''$ where $d \in \mathbb{Z}_p$ is given by g^d using a group generator g and $\boldsymbol{h}''$ are values generated by the simulator. This setting hides the values of $\boldsymbol{h}'$ using $\boldsymbol{h}''$ to the adversary. Also, the simulator enables to project $\boldsymbol{h}'$ using g^d, which is indistinguishable from a random value in the assumption to which the security is reduced.

In our framework, the exponents of public parameters are more complex monomials, but the simulator still can hide HCVs before they are projected into semi-functional space. In our proof, we let the simulator set a non-linear monomial $f_i(\boldsymbol{w})h_j = f_i(\boldsymbol{w})(dh_j' + h_j'') = d \cdot f_i(\boldsymbol{w}) \cdot h_j' + f_i(\boldsymbol{w}) \cdot h_j''$ where $f_i(\boldsymbol{w})$ is a monomial consisting only of SCVs, which are denoted as $\boldsymbol{w}$. In particular, if g^d

Table 2. Comparisons of normal and semi-functional parts in encoding frameworks

		Normal parts	Semi-functional parts
KSGA [14]	Key	$\boldsymbol{k}(\alpha, x, (1, \boldsymbol{h}); \boldsymbol{r})$	$\boldsymbol{k}(\alpha', x, (1, \boldsymbol{h}'); \boldsymbol{r}')$
	CT	$\boldsymbol{c}(y, (1, \boldsymbol{h}); s, \boldsymbol{s})$	$\boldsymbol{c}(y, (1, \boldsymbol{h}'); s', \boldsymbol{s}')$
A [4]	Key	$\boldsymbol{k}(\alpha, x, \boldsymbol{h}; \boldsymbol{r})$	$\boldsymbol{k}(\alpha', x, \boldsymbol{h}'; \boldsymbol{r})$
	CT	$\boldsymbol{c}(y, \boldsymbol{h}; s, \boldsymbol{s})$	$\boldsymbol{c}(y, \boldsymbol{h}'; s, \boldsymbol{s})$
Ours	Key	$\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r})$	$\boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \boldsymbol{r}')$
	CT	$\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})$	$\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); s', \boldsymbol{s}')$

is indistinguishable from a random value (i.e. g^{d+r} where r is a random value), $g^{f_i(\boldsymbol{w})h_j}$ becomes $g^{d \cdot f_i(\boldsymbol{w}) \cdot h'_j} \cdot g^{r \cdot f_i(\boldsymbol{w})h'_j} \cdot g^{f_i(\boldsymbol{w}) \cdot h''_j}$. Hence, $g^{r \cdot f_i(\boldsymbol{w})h'_j}$ can simulate the semi-functional space, where r and h'_j simulates a random variable and a HCV, respectively. $f_i(w)$ appears in the semi-functional space, but its value is the same as that of the normal space as it is defined as SCV.

Refined α Hiding. In our setting, SCVs are not hidden. It means that their projected values in the semi-functional space are identical to their original values as shown in Table 2. Sharing SCVs makes a security proof complex because it means the values must be defined and fixed before receiving any query from the adversary (i.e. when a system sets up). We address this challenge by refining α hiding property of pair encoding framework. We use two oracles which are indistinguishable from each other to simulate the refined α hiding property. In our setting, the oracles output $g^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}$ as an initial instance so that the simulator creates public keys and normal parts of private keys using shared common variables $\boldsymbol{w}$.

It is worth noting that the oracles in the existing techniques [4,14] do not output any value related to common values but only outputs a group generator g as an initial instance. In the pair encoding framework, because the initial instance does not include any public parameters, the α hiding property is proved by selecting public parameters after they obtain the target predicate of the challenge ciphertext (in selective security proof) or the challenge key (in co-selective security proof). However, we observed that, even in selective security proofs, some common variables can be set without using any information about the challenge ciphertext. This makes us use those variables as SCVs. We show that achieving those oracles is feasible by providing new instances.

2 Related Work

Conjunctive schemes of ABE and Identity-based revocation systems were introduced [7,20] to fill the gap between practice and theory. In those schemes, only an identity can be used to revoke users and the other attributes are used to form an access policy. Inner product encryption [8,13,21,22] naturally achieves a non-monotone access structure using polynomials. However, it is well known that expressing a Boolean formula using inner product is inefficient.

A technique to convert encryption schemes in composite order groups into prime order groups were introduced by Lewko [15] using Dual Pairing Vector Spaces (DPVS) [21,22]. However, their conversion technique is not generic and the size of parameters and the amount of computational work required for encryption/decryption increase linearly with the size of vector it uses. Dual System Groups (DSG) [12] were recently introduced by Chen and Wee. They showed that DSG can be utilized to construct a broad range of encryption schemes in prime order groups. Since then, many generic constructions [1,4,11] of encoding schemes in prime order groups have employed DSG except Kim et al.'s work [14]. In Kim et al.'s work, instead of using DSG, they generalized Lewko and Waters' IBE [17] as is done in this paper, but their technique does not cover encryption schemes with non-linear structure.

The compiler for pair encoding in a prime order group is proposed by Attrapadung [5]. In their technique, the common values are defined as a matrix form, which makes the encoding need more structural assumptions. To address this, they redefined the pair encoding to *regular encoding* with additional structural restrictions, which implies the linearity of common values.

Agrawal and Chase also suggested a new way to prove the security of encoding schemes [2]. They proposed a technique where the security of predicate encryption schemes can be proven by showing their encoding satisfy the symbolic property. Namely, if it is shown that the encoding scheme is mapped to a specific format, then the security is proven without any extra efforts. However, the technique still works under the same structural assumptions the pair encoding framework [3] is based on and it is not clear how the symbolic property works with a non-linear structure.

3 Preliminary

3.1 Bilinear Maps

Let $\mathcal{G}$ be a group generator which takes a security parameter λ as input and outputs (p, G_1, G_2, G_T, e), G_1, G_2 and G_T are cyclic groups of prime order p, and $e : G_1 \times G_2 \rightarrow G_T$ is a map such that $e(g^a, h^b) = e(g, h)^{ab}$ for all $g \in G_1$ $h \in G_2$ and $a, b \in \mathbb{Z}_p$ and $e(g, h) \neq 1 \in G_T$ whenever $g \neq 1$ and $h \neq 1$. We assume that the group operations in G_1, G_2 and G_T, as well as the bilinear map e, are all computable in polynomial time with respect to λ. It should be noted that the map e is symmetric if $G_1 = G_2$. If $G_1 \neq G_2$, the map e is asymmetric.

3.2 Non-monotone Access Structure

Definition 1 *(Access Structure)* [10]. *Let $\{P_1, ..., P_n\}$ be a set of parties. A collection $\mathbb{A} \subset 2^{\{P_1,...,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subset C$, then $C \in \mathbb{A}$. A monotone access structure is a monotone collection $\mathbb{A}$ of non-empty* *subsets of* $\{P_1, ..., P_n\}$*, i.e.,* $\mathbb{A} \subset 2^{\{P_1,...,P_n\}} \setminus \{\}$*. The sets in* $\mathbb{A}$ *are called the authorized sets, and the sets not in* $\mathbb{A}$ *are called the unauthorized sets.*

Definition 2 *(Linear Secret-Sharing Schemes (LSSS))* [10]*. A secret sharing scheme* Π *over a set of parties* $\mathcal{P}$ *is called linear (over* $\mathbb{Z}_p$*) if (1) The shares for each party form a vector over* $\mathbb{Z}_p$*. (2) There exists a matrix* A *called the share-generating matrix for* Π*. The matrix* A *has* m *rows and* ℓ *columns. For all* $i = 1, ..., m$*, the* i^{th} *row of* A *is labeled by a party* $\rho(x)$ *(*ρ *is a function from* $\{1, ..., m\}$ *to* $\mathcal{P}$*). When we consider the column vector* $v = (s, r_2, ..., r_\ell)$*, where* $s \in \mathbb{Z}_p$ *is the secret to be shared and* $r_2, ..., r_\ell \in \mathbb{Z}_p$ *are randomly chosen, then* Av *is the vector of* m *shares of the secret* s *according to* Π*. The share* $(Av)_i$ *belongs to party* $\rho(x)$*.*

Moving from Monotone to Non-monotone Access Structures. For a non-monotone access structure, we adopt a technique from Ostrovsky, Sahai and Waters [24]. They assume a family of linear secret sharing schemes $\{\Pi_{\mathbb{A}}\}_{\mathbb{A}\in\mathcal{A}}$ for a set of monotone access structures $\mathbb{A} \in \mathcal{A}$. For each access structure $\mathbb{A} \in \mathcal{A}$, the set of parties $\mathcal{P}$ underlying the access structures has the following properties: The names of the parties may be of two types: either it is normal (like x) or primed (like x'), and if $x \in \mathcal{P}$ then $x' \in \mathcal{P}$ and vice versa. They conceptually associate primed parties as representing the negation of normal parties.

We let $\tilde{\mathcal{P}}$ denote the set of all normal parties in $\mathcal{P}$. For every set $\tilde{S} \subset \tilde{\mathcal{P}}$, $N(\tilde{S}) \subset \mathcal{P}$ is defined by $N(\tilde{S}) = \tilde{S} \cup \{x' | x \in \tilde{P} \setminus \tilde{S}\}$. For each access structure $\mathbb{A} \in \mathcal{A}$ over a set of parties $\mathcal{P}$, a non-monotone access structure $NM(\mathbb{A})$ over the set of parties $\tilde{\mathcal{P}}$ is defined by specifying that $\tilde{S}$ is authorized in $NM(\mathbb{A})$ iff $N(\tilde{S})$ is authorized in $\mathbb{A}$. Therefore, the non-monotone access structure $NM(\mathbb{A})$ will have only normal parties in its access sets. For each access set $X \in NM(\mathbb{A})$, there will be a set in $\mathbb{A}$ that has the elements in X and primed elements for each party not in X. Finally, a family of non-monotone access structures $\tilde{\mathcal{A}}$ is defined by the set of these $NM(\mathbb{A})$ access structures.

3.3 Computational Assumptions

Our compiler needs three simple static assumptions which are also used in [14, 17]. For the following assumptions, we define $\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{R} \mathcal{G}$ and let $f_1 \in G_1$ and $f_2 \in G_2$ be selected randomly.

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

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

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

Assumption 2 (*LW2*). Let $d, t, w \in \mathbb{Z}_p$ be selected randomly. Given

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

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

Assumption 3 (*Decisional Bilinear Diffie-Hellman (DBDH) Assumption*). Let $a, c, d \in \mathbb{Z}_p$ 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 $T_1 \xleftarrow{R} G_T$.

3.4 Predicate Encryption

We adopt the definition of PE and its adaptive security of [3].

Definition of Predicate Encryption [3]**.** A PE for a predicate $R_\kappa : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ consists of Setup, $\mathsf{Encrypt}$, KeyGen and $\mathsf{Decrypt}$ as follows:

- $\mathsf{Setup}(1^\lambda, \kappa) \to (PK, MSK)$: The algorithm takes in a security parameter 1^λ and an index κ which is allocated uniquely for the function R_κ. It outputs a public parameter PK and a master secret key MSK.
- $\mathsf{Encrypt}(x, M, PK) \to CT$: The algorithm takes in an attribute $x \in \mathcal{X}$, a public parameter PK and a plaintext M. It outputs a ciphertext CT.
- $\mathsf{KeyGen}(y, MSK, PK) \to SK$: The algorithm takes in an attribute $y \in \mathcal{Y}$, MSK and PK. It outputs a private key SK.
- $\mathsf{Decrypt}(PK, SK, CT) \to M$: the algorithm takes in SK for y and CT for x. If $R_\kappa(x, y) = 1$, it outputs a message $M \in \mathcal{M}$. Otherwise, it aborts.

$\mathsf{Correctness}$. For all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R_\kappa(x, y) = 1$, if SK is the output of $\mathsf{KeyGen}(y, MSK, PK)$ and CT is the output of $\mathsf{Encrypt}(x, M, PK)$ 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 Predicate Encryption [3]**.** A predicate encryption scheme for a predicate function R_κ 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**: $\mathcal{C}$ runs $\mathsf{Setup}(1^\lambda, \kappa)$ to create (PK, MSK). PK is sent to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ requests a private key for $y_i \in \mathcal{Y}$ and $i \in [q_1]$. For each y_i, $\mathcal{C}$ returns SK_i created by running $\mathsf{KeyGen}(y_i, MSK, PK)$.
- **Challenge**: When $\mathcal{A}$ requests the challenge ciphertext of $x \in \mathcal{X}$, for $R_\kappa(x, y_i) = 0$; $\forall i \in [q_1]$, and submits two messages M_0 and M_1, $\mathcal{C}$ 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 $i = q_1 + 1, ..., q_t$ such that $R_\kappa(x, y_i) = 0$; $\forall i \in \{q_1 + 1, ..., q_t\}$.
- **Guess**: $\mathcal{A}$ outputs $b' \in \{0, 1\}$. If $b = b'$, then $\mathcal{A}$ wins.

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

4 Our Encoding Framework

We introduce our new encoding framework. We largely take a notion of pair encoding framework to describe our encoding. However, our encoding framework can capture the predicate family that has non-linear common variables.

4.1 Syntax

Our encoding scheme for a predicate function R_κ in prime order p consists of four deterministic algorithms Param, Enc_1, Enc_2 and Pair.

- $\mathsf{Param}(\kappa) \rightarrow (\boldsymbol{b} := (b_1, b_2, ..., b_\omega); \omega_1, \omega_2, \omega)$: It takes as input a predicate family κ and outputs integers $\omega_1, \omega_2, \omega \in p$ and a sequence of monomials $\{b_i\}_{i\in[\omega]} \in \mathbb{Z}_p$ with the sequence of variables of $\{b_0, h_j; h_j \in \boldsymbol{h}\}$ and functions f_i where $b_0 \in \mathbb{Z}_p$, $\boldsymbol{h} \in \mathbb{Z}_p^{\omega_2}$ and $f_i(\boldsymbol{w})$ is a monomial consisting of the elements of $\boldsymbol{w} \in \mathbb{Z}_p^{\omega_1}$. That is, for all $i \in [\omega]$, $b_i = b_0 f_i(\boldsymbol{w})$ or $f_i(\boldsymbol{w})h_j$. $\boldsymbol{b}$ shared by the following two algorithms Enc_1 and Enc_2. We let $\boldsymbol{w} = (w_1, ..., w_{\omega_1})$ denote the SCVs and $\boldsymbol{h} = (h_1, ..., h_{\omega_2})$ denote the HCVs.
- $\mathsf{Enc}_1(x \in \mathcal{X}) \rightarrow (\boldsymbol{k} := (k_1, k_2, ..., k_{m_1}); m_2)$: It takes as inputs $x \in \mathcal{X}$ and outputs a sequence of polynomials $\{k_i\}_{i\in[m_1]}$ with coefficients in $\mathbb{Z}_p$, and $m_2 \in \mathbb{Z}_p$ where m_2 is the number of random variables. Every polynomial k_i is a linear combination of monomials of the form $\alpha, r_i b_0, \alpha b_j, r_i b_j$ in variables $\alpha, r_1, ..., r_{m_2}$ and $b_0, b_1, ..., b_\omega$. In more detail, for $i \in [m_1]$,

$$k_i := \delta_i \alpha + \sum\nolimits_{j\in[m_2]} \delta_{i,j} r_j b_0 + \sum\nolimits_{j\in[m_2],k\in[\omega]} \delta_{i,j,k} r_j b_k$$

  where $\delta_i, \delta_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$ are constants which define k_i.
- $\mathsf{Enc}_2(y \in \mathcal{Y}) \rightarrow (\boldsymbol{c} := (c_1, c_2, ..., c_{\tilde{m}_1}); \tilde{m}_2)$: It takes as inputs $y \in \mathcal{Y}$ and outputs a sequence of polynomials $\{c_i\}_{i\in[\tilde{m}_1]}$ with coefficients in $\mathbb{Z}_p$, and $\tilde{m}_2 \in \mathbb{Z}_p$ where $\tilde{m}_2$ is the number of random variables. Every polynomial c_i is a linear combination of monomials of the form $sb_0, s_i b_0, sb_j, s_i b_j$ in variables $s, s_1, ..., s_{\tilde{m}_2}$ and $b_0, b_1, ..., b_\omega$. In more detail, for $i \in [\tilde{m}_1]$,

$$c_i := \phi_i s\, b_0 + \sum\nolimits_{j\in[\tilde{m}_2]} \phi_{i,j} s_j b_0 + \sum\nolimits_{j\in[\tilde{m}_2],k\in[\omega]} \phi_{i,j,k} s_j b_k$$

  where $\phi_i, \phi_{i,j}, \phi_{i,j,k} \in \mathbb{Z}_p$ are constants which define c_i.
- $\mathsf{Pair}(x, y) \rightarrow \boldsymbol{E}$: It takes inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$. It outputs $\boldsymbol{E} \in \mathbb{Z}_p^{m_1\times\tilde{m}_1}$.

Correctness: The correctness holds symbolically when $b_0 = 1$. if $R_\kappa(x, y) = 1$, for every $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R_\kappa(x, y) = 1$, there exists $\boldsymbol{E} \in \mathbb{Z}_p^{m_1\times\tilde{m}_1}$ satisfying $\boldsymbol{k}\boldsymbol{E}\boldsymbol{c}^\top = \alpha s$ where $\boldsymbol{k}\boldsymbol{E}\boldsymbol{c}^\top = \sum_{i\in[m_1],j\in[\tilde{m}_1]} E_{i,j} k_i c_j$.

4.2 Properties

Our encodings satisfy the following properties.

Property 1 (Linearity in hidden common variables). Suppose $\boldsymbol{w}$, $\boldsymbol{r}$, s and $\boldsymbol{s}$ are fixed, our encodings are linear in α and $\boldsymbol{h}$ for all $(\alpha, b_0, \boldsymbol{h}) \in \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p^{\omega_2}$. That is, for all $\alpha, \alpha', b_0, b_0' \in \mathbb{Z}_p, \boldsymbol{h}, \boldsymbol{h}' \in \mathbb{Z}_p^{\omega_2}$, the followings hold:

$$\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{r}) + \boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, b_0', \boldsymbol{h}'); \boldsymbol{r}) = \boldsymbol{k}(\alpha + \alpha', x, \boldsymbol{b}(\boldsymbol{w}, b_0 + b_0', \boldsymbol{h} + \boldsymbol{h}'); \boldsymbol{r})$$

$$\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); s, \boldsymbol{s}) + \boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, b_0', \boldsymbol{h}'); s, \boldsymbol{s}) = \boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, b_0 + b_0', \boldsymbol{h} + \boldsymbol{h}'); s, \boldsymbol{s})$$

Property 2 (Linearity in random variables). Suppose $\boldsymbol{w}$ and $\boldsymbol{h}$ are fixed, our encodings are linear in α, s, $\boldsymbol{r}$ and $\boldsymbol{s}$ for all $(\alpha, s, \boldsymbol{r}, \boldsymbol{s}) \in \mathbb{Z}_p \times \mathbb{Z}_p \times \mathbb{Z}_p^{m_2} \times \mathbb{Z}_p^{\tilde{m}_2}$. That is, for all $\alpha, \alpha', s, s' \in \mathbb{Z}_p, \boldsymbol{r}, \boldsymbol{r}' \in \mathbb{Z}_p^{\tilde{m}_2}$ and $\boldsymbol{s}, \boldsymbol{s}' \in \mathbb{Z}_p^{\tilde{m}_2}$, the followings hold:

$$k(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{r}) + k(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{r}') = k(\alpha + \alpha', x, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{r} + \boldsymbol{r}')$$

$$c(y, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{s}) + c(y, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{s}') = c(y, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{s} + \boldsymbol{s}')$$

where $\boldsymbol{w}, b_0, \boldsymbol{h} \in \mathbb{Z}_p^{\omega_1} \times \mathbb{Z}_p \times \mathbb{Z}_p^{\omega_2}$.

Property 3 (Parameter Vanishing). For all $\alpha, b_0, b_0' \in \mathbb{Z}_p, \boldsymbol{w}, \boldsymbol{w}' \in \mathbb{Z}_p^{\omega_1}, \boldsymbol{h}, \boldsymbol{h}' \in \mathbb{Z}_p^{\omega_2}$, there exists $\mathbf{0} \in \mathbb{Z}_p^{2k+1}$ which makes the distributions of $\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \mathbf{0})$ and $\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}', b_0', \boldsymbol{h}'); \mathbf{0})$ are statistically identical.

Property 4 (α hiding). We let $g_1 \xleftarrow{R} G_1$, $g_2 \xleftarrow{R} G_2$, $\alpha, s \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{w} \xleftarrow{R} \mathbb{Z}_p^{\omega_1}$, $\boldsymbol{h} \xleftarrow{R} \mathbb{Z}_p^{\omega_2}, \boldsymbol{r} \xleftarrow{R} \mathbb{Z}_p^{w_2}$ and $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_p^{m_2}$. For all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R_\kappa(x, y) = 0$, the following two distributions are indistinguishable:

$$\{g_1^{\boldsymbol{b}(\boldsymbol{w},1,\mathbf{1})}, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\mathbf{1})}, g_1^{\boldsymbol{c}(y,(\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}, g_2^{\boldsymbol{k}(\alpha,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})}\}$$

$$\approx \{g_1^{\boldsymbol{b}(\boldsymbol{w},1,\mathbf{1})}, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\mathbf{1})}, g_1^{\boldsymbol{c}(y,(\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}, g_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})}\}.$$

4.3 The Compiler

For a predicate family $R_\kappa : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ and its encoding $E(R_\kappa, p)$, A PE scheme $PE(E(R_\kappa, p))$ consists of four algorithms **Setup**, **KeyGen**, **Encrypt** and **Decrypt**.

- **Setup**$(1^\lambda, \kappa) \to \langle PK, MSK \rangle$. The setup algorithm randomly chooses bilinear groups $\mathcal{G} = (p, G_1, G_2, G_T, e)$ of prime order $p > 2^\lambda$. It takes group generators $g_1 \xleftarrow{R} G_1, g_2 \xleftarrow{R} G_2$ from $\mathcal{G}$. It executes $(\boldsymbol{b}, \omega_1, \omega_2, \omega) \leftarrow \mathsf{Param}$ and sets $b_0 = 1$. It randomly selects $\alpha, a, y_u, y_v, y_f \in \mathbb{Z}_p$, $\boldsymbol{w} \in \mathbb{Z}_p^{\omega_1}$ and $\boldsymbol{h} \in \mathbb{Z}_p^{\omega_2}$. It sets $\tau = y_v + a \cdot y_u$. It publishes public parameters (PK) as

$$\{e(g_1, g_2)^\alpha, g_1, g_1^a, g_1^\tau, g_1^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, g_1^{a \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, g_1^{\tau \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}\}.$$

  It sets MSK as $\{\alpha, g_2, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, f_2 = g_2^{y_f}, u_2 = f_2^{y_u}, v_2 = f_2^{y_v}\}$.

- **KeyGen**$(x, MSK) \rightarrow SK$. The algorithm takes as inputs $x \in \mathcal{X}$ and MSK. To generate SK, it runs $(\boldsymbol{k}; m_2) \leftarrow \mathsf{Enc}_1$ and randomly selects $\boldsymbol{r} \in \mathbb{Z}_p^{m_2}$ and $\boldsymbol{z} \in \mathbb{Z}_p^{|\boldsymbol{k}|}$. It parses α from MSK and outputs $SK := (\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3)$ where $\boldsymbol{D}_1 = g_2^{\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r})} v_2^{\boldsymbol{z}}$, $\boldsymbol{D}_2 = u_2^{\boldsymbol{z}}$, $\boldsymbol{D}_3 = f_2^{-\boldsymbol{z}}$.
- **Encrypt**$(M, y, PK) \rightarrow CT$. The algorithm takes as inputs $y \in \mathcal{Y}$, a message M and PK. It runs $(\boldsymbol{c}; \tilde{m}_2) \leftarrow \mathsf{Enc}_2$ and randomly selects $s \in \mathbb{Z}_p$ and $\boldsymbol{s} \in \mathbb{Z}_p^{\tilde{m}_2+1}$. The algorithm sets $C_0 = M \cdot e(g_1, g_2)^{\alpha s}$ and outputs $CT := (C_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$ where $\boldsymbol{C}_1 = g_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})}$, $\boldsymbol{C}_2 = (g_1^a)^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})}$, $\boldsymbol{C}_3 = (g_1^{\tau})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})}$.
- **Decrypt**$(x, y, SK, CT) \rightarrow M$. It takes as inputs SK for $x \in \mathcal{X}$ and CT for $y \in \mathcal{Y}$. It runs $\boldsymbol{E} \leftarrow \mathsf{Pair}(x, y)$ and computes

$$A_1 = e(\boldsymbol{C}_1^{\boldsymbol{E}^\top}, \boldsymbol{D}_1),\ A_2 = e(\boldsymbol{C}_2^{\boldsymbol{E}^\top},\ \boldsymbol{D}_2), A_3 = e(\boldsymbol{C}_3^{\boldsymbol{E}^\top}, \boldsymbol{D}_3).$$

  Suppose $R_\kappa(x, y) = 1$, $A_1 \cdot A_2 \cdot A_3 = e(g_1, g_2)^{\alpha s}$. It outputs $M = C_0 / e(g_1, g_2)^{\alpha s}$.

Correctness. For $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R_\kappa(x, y) = 1$, $\boldsymbol{E}$ is a reconstruction matrix such that $\boldsymbol{c}\boldsymbol{E}^\top \boldsymbol{k}^\top = \alpha s$ when $b_0 = 1$. Hence, we can compute followings:

$$A_1 = e(\boldsymbol{C}_1^{\boldsymbol{E}^\top}, \boldsymbol{D}_1) = e(g_1, g_2)^{\boldsymbol{c}\boldsymbol{E}^\top \boldsymbol{k}^\top} e(g_1, v_2)^{\boldsymbol{c}\boldsymbol{E}^\top \boldsymbol{z}^\top} = e(g_1, g_2)^{\alpha s} e(g_1, v_2)^{\boldsymbol{c}\boldsymbol{E}^\top \boldsymbol{z}^\top}$$

$$A_2 = e(\boldsymbol{C}_2^{\boldsymbol{E}^\top}, \boldsymbol{D}_2) = e(g_1, u_2)^{a \cdot \boldsymbol{c}\boldsymbol{E}^\top \boldsymbol{z}^\top},\ A_3 = e(\boldsymbol{C}_3^{\boldsymbol{E}^\top}, \boldsymbol{D}_3) = e(g_1, f_2)^{-\tau \cdot \boldsymbol{c}\boldsymbol{E}^\top \boldsymbol{z}^\top}$$

It should be noted that $\tau = y_v + a y_u$ where y_v and y_u are discrete logarithms of v_2 and u_2 to the base f_2, respectively. Therefore, $A_1 \cdot A_2 \cdot A_3 = e(g_1, g_2)^{\alpha s}$.

Theorem 1. *Suppose the assumptions LW1, LW2 and DBDH hold in $\mathcal{G}$, for all encoding $E(R_\kappa, p)$ with a predicate family R_κ and a prime p, $PE(E(R_\kappa, p))$ is adaptively secure. Precisely, for any PPT adversary $\mathcal{A}$, there exist PPT algorithms $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$ and $\mathcal{B}_4$, whose running times are the same as $\mathcal{A}$ such that, for any λ,*

$$Adv_{\mathcal{A}}^{FE(P)}(\lambda) \leq w_t \cdot Adv_{\mathcal{B}_1}^{LW1}(\lambda) + 2 \cdot m_t \cdot Adv_{\mathcal{B}_2}^{LW2}(\lambda) + Adv_{\mathcal{B}_3}^{DBDH}(\lambda) + q \cdot Adv_{\mathcal{B}_4}^{\alpha\text{-}hd}(\lambda)$$

where (1) q is the number of key queries in phases I/II, (2) m_t is the total number of random variables used to simulate all private keys, (3) w_t is the number of random variables used in the challenge ciphertext and (4) $Adv_{\mathcal{B}_4}^{\alpha\text{-}hd}(\lambda)$ is the advantage of $\mathcal{B}_4$ to breaking α hiding.

5 Security Analysis

We define the semi-functional (SF) algorithms for the security analysis. We let the simulator randomly select $\boldsymbol{h}' \in \mathbb{Z}_p^{\omega_2}$.

SFKeyGen$(x, MSK, \boldsymbol{h}', j, \alpha') \rightarrow SK$. The algorithm takes as inputs the master secret key MSK, $x \in \mathcal{X}$ and $j \in \{0, ..., m_2\}$. Then, the algorithm selects $\alpha' \xleftarrow{R}$

$\mathbb{Z}_p$ and $\tilde{\boldsymbol{r}}_j \xleftarrow{R} \mathbb{Z}_p^{m_2}$ of which the first j elements are random variables and the others are 0. It also creates a normal key $(\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3)$ using **KeyGen**. It outputs $SK := \langle \boldsymbol{D}'_1, \boldsymbol{D}'_2, \boldsymbol{D}'_3 \rangle$ where $\boldsymbol{D}'_1 = \boldsymbol{D}_1 \cdot f_2^{-a\boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{\boldsymbol{r}}_j)}, \boldsymbol{D}'_2 = \boldsymbol{D}_2 \cdot f_2^{-\tau \boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{\boldsymbol{r}}_j)}, \boldsymbol{D}'_3 = \boldsymbol{D}_3$. We define the type of SK as follows:

$$\text{The type of SK} : \begin{cases} \text{Nominally semi-functional (NSF)} & \text{if } \alpha' = 0 \\ \text{Temporary semi-functional (TSF)} & \text{if } \alpha' \neq 0 \text{ and } j \neq 0 \\ \text{Semi-functional (SF)} & \text{if } \alpha' \neq 0 \text{ and } j = 0 \end{cases}$$

In SF keys, $\tilde{\boldsymbol{r}}_0$ equals to the zero vector $\boldsymbol{0}$ by the definition. Due to the parameter vanishing property, we can rewrite SF keys (SF-SK) as follows:

$$\boldsymbol{D}'_1 = \boldsymbol{D}_1 \cdot f_2^{-a\boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 0, \boldsymbol{0}); \boldsymbol{0})}, \boldsymbol{D}'_2 = \boldsymbol{D}_2 \cdot f_2^{-\tau \boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 0, \boldsymbol{0}); \boldsymbol{0})}.$$

SFEncrypt$(M, y, PK, \boldsymbol{h}', j) \rightarrow CT$. The algorithm takes as inputs a message M, the public key PK and a description $y \in \mathcal{Y}$ and $j \in [\tilde{m}_2 + 1]$. It sets $f_1 = g_1^{y_f}$ and $u_1 = f_1^{y_u}$. It generates a normal ciphertext $(C_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$. If $j = 1$, it selects $\tilde{s} \xleftarrow{R} \mathbb{Z}_p$. The algorithm sets $C'_0 = C_0$ and outputs CT following:

$$\boldsymbol{C}'_1 = \boldsymbol{C}_1,\ \boldsymbol{C}'_2 = \boldsymbol{C}_2 \cdot f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{s}, \boldsymbol{0})},\ \boldsymbol{C}'_3 = \boldsymbol{C}_3 \cdot u_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{s}, \boldsymbol{0})}.$$

If $j > 1$, it selects a random value $\tilde{s} \xleftarrow{R} \mathbb{Z}_p$ and a random vector $\tilde{\boldsymbol{s}}_{j-1} \xleftarrow{R} \mathbb{Z}_p^{\tilde{m}_2}$ where the first $j-1$ elements are random variables and the others are 0. The algorithm then sets $C'_0 = C_0$ and outputs $CT := \langle C'_0, \boldsymbol{C}'_1, \boldsymbol{C}'_2, \boldsymbol{C}'_3 \rangle$ where

$$\boldsymbol{C}'_1 = \boldsymbol{C}_1,\ \boldsymbol{C}'_2 = \boldsymbol{C}_2 \cdot f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{s}, \tilde{\boldsymbol{s}}_{j-1})},\ \boldsymbol{C}'_3 = \boldsymbol{C}_3 \cdot u_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{s}, \tilde{\boldsymbol{s}}_{j-1})}.$$

In particular, we call CT a semi-functional (SF) ciphertext if $j = \tilde{m}_2 + 1$.

We summarize the security games that we use for the security proof in Table 3. In the proof, we will show that all games in Table 3 are indistinguishable. The most critical proof among them is the invariance between games $\mathsf{G}_{k,j-1}^N$ and $\mathsf{G}_{k,j}^N$ where $j \in [m_2]$. This shows how we feature the jth random variable in the normal space to the semi-functional space. We provide this proof in Lemma 2. We will show the other proofs (of Lemmas 1, 3, 4 and 5) in the full version of this paper.

Lemma 1. Suppose there exists a PPT $\mathcal{A}$ who can distinguish $\mathsf{G}_{0,i}$ and $\mathsf{G}_{0,i+1}$ with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks $LW1$ with the advantage ϵ using $\mathcal{A}$.

Lemma 2. Suppose there exists a PPT $\mathcal{A}$ who can distinguish $\mathsf{G}_{k,j-1}^N$ and $\mathsf{G}_{k,j}^N$ for $j \in [m_2]$ with non-negligible advantage ϵ where m_2 is the size of random variables that the kth key uses. Then, we can build an algorithm $\mathcal{B}$ which breaks $LW2$ with the advantage ϵ using $\mathcal{A}$.

Table 3. Games for security analysis

G_{Real}	: This is a real game that all keys and ciphertexts are normal. (= $\mathsf{G}_{0,0}$)
$\mathsf{G}_{0,j}$	: $CT \leftarrow \textbf{SFEncrypt}(M, y, PK, \boldsymbol{h}', j)$ for $j = 1, ..., \tilde{m}_2 + 1$
G_0	: (= $\mathsf{G}_{0,\tilde{m}_2+1} = \mathsf{G}^N_{1,0}$ by the definitions)
$\mathsf{G}^N_{k,j}$	: $(k \geq 1)$ $\alpha'_i \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{h}' \xleftarrow{R} \mathbb{Z}_p^{\omega_2}$ $SK_i \leftarrow \begin{cases} \textbf{SFKeyGen}(x, MSK, \boldsymbol{0}, 0, \alpha'_i) & \text{if } i < k \text{ (type = SF)} \\ \textbf{SFKeyGen}(x, MSK, \boldsymbol{h}', j, \boxed{0}) & \text{if } i = k \text{ (type = NSF)} \\ \textbf{KeyGen}(x, MSK) & \text{if } i > k \text{ (type = Normal)} \end{cases}$
G^T_{k,m_2-j}	: $(k \geq 1)$ $\alpha'_i \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{h}' \xleftarrow{R} \mathbb{Z}_p^{\omega_2}$ $SK_i \leftarrow \begin{cases} \textbf{SFKeyGen}(x, MSK, \boldsymbol{0}, 0, \alpha'_i) & \text{if } i < k \text{ (type = SF)} \\ \textbf{SFKeyGen}(x, MSK, \boldsymbol{h}', m_2 - j, \boxed{\alpha'_i}) & \text{if } i = k \text{ (type = TSF)} \\ \textbf{KeyGen}(x, MSK) & \text{if } i > k \text{ (type = Normal)} \end{cases}$
G_k	: $(k \geq 1)$ (= $\mathsf{G}^T_{k,0} = \mathsf{G}^N_{k+1,0}$ by the definitions) $\alpha'_i \xleftarrow{R} \mathbb{Z}_p$, $SK_i \leftarrow \begin{cases} \textbf{SFKeyGen}(x, MSK, \boldsymbol{h}', 0, \alpha'_i) & \text{if } i <= k \text{ (type = SF)} \\ \textbf{KeyGen}(x, MSK) & \text{if } i > k \text{ (type = Normal)} \end{cases}$
G_{Final}	: $M' \xleftarrow{R} \mathcal{M}$, $CT \leftarrow \textbf{SFEncrypt}(M, y, PK, \boldsymbol{h}', j)$

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 $\alpha \in \mathbb{Z}_p, a, y'_v \in \mathbb{Z}_p, \boldsymbol{w} \in \mathbb{Z}_p^{\omega_1}, \boldsymbol{h}', \boldsymbol{h}'' \in \mathbb{Z}_p^{\omega_2}$. It implicitly sets $y_v = d - aw + y'_v$, $y_u = w$, $b = 1/d$ and $\tau = d - aw + y'_v + aw = d + y'_v$. It sets a public key PK and MSK as follows:

$$PK =: \{e(g_1, g_2)^\alpha = e(f_1^d, f_2^d)^\alpha, g_1 = f_1^d,$$

$$g_1^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = (f_1^d)^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')} f_1^{\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'')}, g_1^a, g_1^{a \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, g_1^\tau = f_1^{d^2}(f_1^d)^{y'_v},$$

$$g_1^{\tau \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = (f_1^{d^2})^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')}(f_1^d)^{\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'')}(f_1^d)^{y'_v \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')}(f_1)^{y'_v \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'')}\}.$$

$$MSK := \{g_2 = f_2^d, g_2^\alpha = (f_2^d)^\alpha, \; g_2^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = (f_2^d)^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')} f_2^{\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'')},$$

$$v_2 = f_2^d (f_2^w)^{-a} f_2^{y'_v}, \; u_2 = f_2^w, \; f_2\}.$$

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

$$\begin{aligned}\boldsymbol{K}_0 =&(f_2^d)^{\boldsymbol{k}(\alpha,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'')} f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');\boldsymbol{r}'')} (f_2^c)^{-\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');\boldsymbol{1}_j)}\\ &\cdot (f_2^d (f_2^w)^{-a} f_2^{y'_v})^{\boldsymbol{z}'} T^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{1}_j)} (f_2^c)^{y'_v \boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\cdot\boldsymbol{1}_j)}\\ &\cdot f_2^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})},\\ \boldsymbol{K}_1 =&(f_2^w)^{\boldsymbol{z}'} T^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{1}_j)} f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})},\\ \boldsymbol{K}_2 =&f_2^{-\boldsymbol{z}'} (f_2^c)^{-\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'),\boldsymbol{1}_j)}\end{aligned}$$

If $T = f_2^{cw}$, then this key is a properly distributed nominally semi-function (NSF) key created using **SFKeyGen**$(x, MSK, \boldsymbol{h}', j-1, 0)$ because

$$\begin{aligned}\boldsymbol{K}_0 = & f_2^{d\cdot\boldsymbol{k}(\alpha',x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'')} \boxed{f_2^{d\cdot\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');-c\cdot\boldsymbol{1}_j)}} f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');\boldsymbol{r}'')}\\ &\cdot f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');-c\cdot\boldsymbol{1}_j)} f_2^{(d-wa+y'_v)(\boldsymbol{z}')} \boxed{f_2^{d\cdot\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');c\cdot\boldsymbol{1}_j)}}\\ &\cdot f_2^{-wa\cdot\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');c\cdot\boldsymbol{1}_j)} f_2^{y'_v\cdot\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');c\cdot\boldsymbol{1}_j)} f_2^{-a\cdot\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})} \quad (1)\end{aligned}$$

$$\begin{aligned}&= f_2^{d\boldsymbol{k}(\alpha',x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r})} f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');\boldsymbol{r})} f_2^{(d-wa+y'_v)(\boldsymbol{z}'+\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');c\cdot\boldsymbol{1}_j))}\\ &\quad\cdot f_2^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})} \quad (2)\\ &= f_2^{\boldsymbol{k}(d\alpha',x,\boldsymbol{b}(\boldsymbol{w},d,d\boldsymbol{h}'+\boldsymbol{h}'');\boldsymbol{r})} f_2^{(d-wa+y'_v)(\boldsymbol{z}'+\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');c\cdot\boldsymbol{1}_j))}\\ &\quad\cdot f_2^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})} \quad (3)\\ &= g_2^{\boldsymbol{k}(\alpha',x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})} v_2^{\boldsymbol{z}} f_2^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})}\end{aligned}$$

$$\boldsymbol{K}_1 = (f_2^w)^{\boldsymbol{z}'} (f_2^{cw})^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{1}_j)} f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})} = u_2^{\boldsymbol{z}} f_2^{\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})}$$

This implicitly sets $\boldsymbol{r} = \boldsymbol{r}'' - c \cdot \boldsymbol{1}_j$ and $\boldsymbol{z} = \boldsymbol{z}' + \boldsymbol{k}(0, x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); c \cdot \boldsymbol{1}_j)$. The equality (1) in above equation holds by the *linearity in random values*. The equality (2) holds because of the definition of $\boldsymbol{r}$ $(= \boldsymbol{r}'' - c \cdot \boldsymbol{1}_j)$ and linearity in random values. The equality (3) holds due to linearity in hidden common variables.

Otherwise, if T is a random and we let $f_2^{cw+\gamma}$ denote T, this is also a properly distributed (NSF) key but it was created using **SFKeyGen**$(x, MSK, \boldsymbol{h}', j, 0)$ since this implicitly sets $\boldsymbol{r}'_j = \boldsymbol{r}'_{j-1} + \gamma \cdot \boldsymbol{1}_j$. It is worth noting that $\boldsymbol{r}'_j$ is uniformly random because γ is randomly distributed.

Challenge: When the adversary requests the challenge ciphertext with two messages 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 $\boldsymbol{s}'', \tilde{\boldsymbol{s}} \in \mathcal{R}_s$. Then, it implicitly sets $s = wt\tilde{s} + s''$, $s' = -d^2t\tilde{s}$, $\boldsymbol{s}' = wt\tilde{\boldsymbol{s}} + \boldsymbol{s}''$ and $\boldsymbol{s}' = -d^2t\tilde{\boldsymbol{s}}$. Because of s'', $\tilde{s}$, $\boldsymbol{s}''$ and $\tilde{\boldsymbol{s}}$, 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

$$\begin{aligned}
\boldsymbol{C}_0 &= (f_1^{dwt})^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{s},\tilde{\boldsymbol{s}})}(f_1^{d})^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s'',\boldsymbol{s}'')}(f_1^{wt})^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');\tilde{s},\tilde{\boldsymbol{s}})} \\
&\quad \cdot f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');s'',\boldsymbol{s}'')} \\
&= g_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})} \\
\boldsymbol{C}_1 &= (C_0)^a (f_1^{d^2t})^{-\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{s},\tilde{\boldsymbol{s}})} = g_1^{a\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})} f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s',\boldsymbol{s}')} \\
\boldsymbol{C}_2 &= (f_1^{d^2})^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s'',\boldsymbol{s}'')}(f_1^{dwt})^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},y'_v,\boldsymbol{h}''+y'_v\boldsymbol{h}');\tilde{s},\tilde{\boldsymbol{s}})} \\
&\quad \cdot (f_1^{d})^{c(y,\boldsymbol{b}(\boldsymbol{w},y'_v,\boldsymbol{h}''+y'_v\boldsymbol{h}');s'',\boldsymbol{s}'')}(f_1^{wt})^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,y'_v\boldsymbol{h}'');\tilde{s},\tilde{\boldsymbol{s}})} f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,y'_v\boldsymbol{h}'');s'',\boldsymbol{s}'')} \\
&= g_1^{\tau\cdot\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})} u_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s',\boldsymbol{s}')}.
\end{aligned}$$

Therefore, the challenge ciphertext is properly distributed. The equalities in the above equations hold by both *linearity in hidden common variables* and *linearity in random values*. In particular, the last equalities in $\boldsymbol{C}_0$, $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ hold because of $s' = -d^2t\tilde{s}$, $\boldsymbol{s}' = -d^2t\tilde{\boldsymbol{s}}$ and the definitions of public parameters. $\tilde{s}$ and $\tilde{\boldsymbol{s}}$ are randomly distributed to the adversary although they also appear in $s = wt\tilde{s} + s''$, $\boldsymbol{s} = wt\tilde{\boldsymbol{s}} + \boldsymbol{s}''$ since their values are not revealed due to s'' and $\boldsymbol{s}''$, which are uniquely allocated random values. $\square$

Lemma 3. Suppose there exists an $\mathcal{A}$ who can distinguish G^N_{k,m_2} and G^T_{k,m_2} with non-negligible advantage ϵ for any $k < q$. Then, we can build an algorithm $\mathcal{B}$ who can break the α hiding property with ϵ using $\mathcal{A}$.

Lemma 4. Suppose there exists a PPT $\mathcal{A}$ who can distinguish $\mathsf{G}^T_{k,j-1}$ and $\mathsf{G}^T_{k,j}$ for $j \in [m_2]$ with non-negligible advantage ϵ where m_2 is the size of random variables that the kth key uses. Then, we can build an algorithm $\mathcal{B}$ which breaks $LW2$ with the advantage ϵ using $\mathcal{A}$.

Lemma 5. Suppose there exists a PPT $\mathcal{A}$ who can distinguish G_{q_t} and G_{Final} with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks $DBDH$ with the advantage ϵ using $\mathcal{A}$.

6 Adaptively Secure NM-CP-ABE with Short Keys

We introduce an NM-CP-ABE with short keys. The part of the security proof, co-selective security, is inspired by the selective NM-KP-ABE scheme of [6].

Assumptions for NM-CP-ABE with Short Keys. We define two computational assumptions in an asymmetric pairing. We take (n-A2) from [26] and use n-DBDHE. We modify them to prove α hiding using the technique that Lewko and Waters introduced in [19]. We provide the security of our assumptions in the generic group model in the full version of this paper.

Assumption 4 *(n-A2).* If a group generator $\mathcal{G}$ and a positive integer n are given, we define the following distribution

$$\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{R} \mathcal{G}, \quad c, d, a, b_1, ..., b_n \xleftarrow{R} \mathbb{Z}_p,$$

$$g_1 \xleftarrow{R} G_1, \quad g_2 \xleftarrow{R} G_2, \quad D := \{g_1, g_2, g_1^c, g_2^c\} \cup \{g_1^{z_1}, g_2^{z_2} | z_1 \in Z_1, z_2 \in Z_2\}$$

$$\begin{aligned}
Z_1 = \{ \quad & \forall (i,j) \in [n,n],\ dc, a, b_j, dcb_j, dcb_ib_j, a^i/b_j^2 \\
& \forall (i,j,j') \in [2n,n,n], j \neq j',\ a^ib_j/b_{j'}^2 \\
& \forall (i,j,j') \in [n,n,n], j \neq j',\ dca^ib_j/b_{j'}, dca^ib_j/b_{j'}^2 \\
& \forall (i,j,j',j'') \in [n,n,n,n], j \neq j', j' \neq j''\},\ dca^ib_jb_{j'}/b_{j''}^2 \quad \}, \\
Z_2 = \{ \quad & \forall (i,j) \in [n,n],\ dc, a^i, a^ib_j, a^i/b_j^2 \\
& \forall (i,j) \in [2n,n], i \neq n+1,\ a^i/b_j \\
& \forall (i,j,j') \in [2n,n,n], j \neq j',\ a^ib_j/b_{j'}^2 \quad \}.
\end{aligned}$$

Given the instances, distinguishing between $T_0 = g_2^{da^{n+1}}$ and $T_1 \xleftarrow{R} G_2$ is hard.

Assumption 5 *($n-DBDHE$).* If a group generator $\mathcal{G}$ and a positive integer n are given, we define the following distribution

$$\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{R} \mathcal{G},\ b, c, d, \xleftarrow{R} \mathbb{Z}_p,$$

$$g_1 \xleftarrow{R} G_1,\ g_2 \xleftarrow{R} G_2,\ D := \{g_1, g_2, g_1^c, g_2^c\} \cup \{g_1^{z_1}, g_2^{z_2} | z_1 \in Z_1, z_2 \in Z_2\}$$

where $Z_1 = Z_2 := \{dc, b^i |\ \forall i \in [2n], i \neq n+1\}$.
Given D, it is hard to distinguish between $T_0 = g_2^{db^{n+1}}$ and $T_1 \xleftarrow{R} G_2$.

We define the advantage of an algorithm $\mathcal{A}$ to break *n-A2* or *n-DBDHE* as

$$Adv_{\mathcal{G},\mathcal{A},n}^{\{A2,\ DBDHE\}}(\lambda) = |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|$$

Encoding Scheme for NM-CP-ABE with Short Keys. Our encoding scheme for NM-CP-ABE with short keys consists of the following encoding algorithms:

- **Param**(κ): It sets $\omega_1 = 1, \omega_2 = 2N+3$ and $\omega = 3N+4$. It selects $\alpha \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{w} = \eta \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{h} = (\delta, \nu, \zeta, y_1, ..., y_N, y_1', ..., y_N') \xleftarrow{R} \mathbb{Z}_p^{2N+3}$. It sets $\boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}) = (\delta,\ \nu,\ \zeta,\ \eta,\ y_1,\ ..., y_N,\ y_1',\ ..., y_N',\ \eta \cdot y_1', ..., \eta \cdot y_N')$.
- **Enc**$_1(S)$: The algorithm selects $r_0, r_1, r_2 \xleftarrow{R} \mathbb{Z}_p$ and sets $\boldsymbol{r} = (r_0, r_1, r_2)$. It sets $d_1 = \alpha + \delta r_2 + \nu r_0$, $d_2 = -r_0$, $d_3 = r_2$. For all $w_i \in S = \{w_1, ..., w_k\}$ such that S is not an empty set and $k \leq N$. It sets

$$d_4 = -\zeta r_2 + (y_1 a_1 + ... + y_N a_N) r_1, \quad d_5 = r_1,$$

$$d_6' = \eta (y_1' a_1 + ... + y_N' a_N) r_2, \quad d_7' = \eta r_2$$

  where a_i is an coefficient of z^{i-1} in $P(z) = \prod_{w \in S}(z - w)$ for $i \in [k+1]$. It defines $\boldsymbol{k}(\alpha, S, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r}) := (d_1, d_2, d_3, d_4, d_5, d_6', d_7')$.
- **Enc**$_2(\tilde{\mathbb{A}})$: For the non-monotone access structure $\tilde{\mathbb{A}}$, there exists a monotone access structure $\tilde{\mathbb{A}} = NM(\mathbb{A})$ where $\mathbb{A} = (A, \rho)$ and A is an $\ell \times m$ access matrix. The algorithm randomly selects $s, s_2, ..., s_m, t_1, ..., t_\ell \xleftarrow{R} \mathbb{Z}_p$ and sets $\boldsymbol{s} = (s_2, ..., s_m, t_1, ..., t_\ell)$ and $\lambda_i = A_i \cdot \boldsymbol{\phi}$ where A_i is the ith row

of A and $\boldsymbol{\phi} = (s, s_2, ..., s_m)$. It sets $c_1 = s$, $c_2 = \nu s$. For all $i \in [\ell]$, it sets $\boldsymbol{c}(\tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s}) := (c_1, c_2, c_{i,1}, c_{i,2}, ..., c_{i,N+2}; \forall i \in [\ell])$ as follows:

$$c_{i,1} = \delta\lambda_i + \zeta t_i, \quad c_{i,2} = t_i,$$
$$c_{i,3} = -(y_2 - y_1\rho(i))t_i, \quad ..., \quad c_{i,N+1} = -(y_N - y_1\rho(i)^{N-1})t_i \quad \text{if } \rho(i) = x_i;$$
$$c_{i,1} = \delta\lambda_i - \eta y'_1 t_i, \quad c_{i,2} = t_i,$$
$$c_{i,3} = -(y'_2 - y'_1\rho(i))t_i, \quad ..., \quad c_{i,N+1} = -(y'_N - y'_1\rho(i)^{N-1})t_i \quad \text{if } \rho(i) = x'_i;$$

where the attribute corresponding to the ith row of A by the mapping ρ is denoted by x_i (or x'_i, if it is a negated attribute).

- **Pair**$(S, \tilde{\mathbb{A}})$: If S satisfies $\tilde{\mathbb{A}}$, there exists $S' = N(S)$ which satisfies an access structure $\mathbb{A} = (A, \rho)$ such that $\tilde{\mathbb{A}} = NM(\mathbb{A})$. We define $I = \{i|\rho(i) \in S'\}$. It computes $\boldsymbol{\mu} = (\mu_1, ..., \mu_{|I|})$ such that $\boldsymbol{\mu} \cdot A_I = (1, 0, ..., 0)$. We set γ the index such that $w_\gamma = x_i$. To compute the share of $i \in I$, for $\Lambda_{i\in I}\, \forall i \in I$, it computes $a_0, ..., a_N$ which are the coefficient of z^i in $P(z)$. Then, it sets

$$\Lambda_i = c_{i,1} \cdot d_3 + c_{i,2} \cdot d_4 + \Sigma_{j\in[N]\setminus\{1\}} a_j \cdot c_{i,1+j} \cdot d_5 = \lambda_i \delta r_2 \qquad \text{if } \rho(i) = x_i;$$

$$\Lambda_i = c_{i,1} \cdot d_3 + \frac{c_{i,2} \cdot d'_6 + \Sigma_{j\in[N]\setminus\{1\}} a_j \cdot c_{i,1+j} \cdot d'_7}{\Sigma_{j\in[N]} a_j \cdot \rho(i)^j} = \lambda_i \delta r_2 \quad \text{if } \rho(i) = x'_i.$$

Finally, the algorithm computes $c_1 \cdot d_1 + c_2 \cdot d_2 - \prod_{i\in[I]} \mu_i \Lambda_i = \alpha s$.

We computationally prove the α hiding of our scheme by Lemma 6.

Lemma 6. *Suppose there exists a PPT adversary $\mathcal{A}$ who can break α hiding with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ breaking $n_1 - DBDHE$ or $n_2 - A2$ with ϵ using $\mathcal{A}$ with an attributes set of size $k < n_1, n_2$.*

Proof: We provide this proof in the full version of this paper. □

6.1 Duality

We introduce NM-KP-ABE with short ciphertexts as a dual scheme of our NM-CP-ABE with short keys using the conversion technique in [9]. The following encoding scheme constructs NM-KP-ABE with short ciphertexts:

- Param(κ): It runs Param of NM-CP-ABE to get $\boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h})$ and outputs $\boldsymbol{b}'(\boldsymbol{w}', 1, \boldsymbol{h}') := (\pi, b'(\boldsymbol{w}, 1, \boldsymbol{h}))$ where $\pi \xleftarrow{R} \mathbb{Z}_p$. This sets $\boldsymbol{w}' = \boldsymbol{w}$ and $\boldsymbol{h}' = (\pi, \boldsymbol{h})$.
- Enc$_1(\tilde{\mathbb{A}})$: It runs Enc$_2(\tilde{\mathbb{A}})$ of NM-CP-ABE to get $c(\tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})$ and sets $d'_1 = \alpha + \pi s$ and $\boldsymbol{k}'(\alpha, \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}', 1, \boldsymbol{h}'); \boldsymbol{r}') := (d'_1, c(\tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s}))$. It is worth noting that s can be parsed from $\boldsymbol{c}$. It implicitly sets $\boldsymbol{r}' = (s, \boldsymbol{s})$.
- Enc$_2(S)$: It creates $s' \xleftarrow{R} \mathbb{Z}_p$ and runs Enc$_1(S)$ of NM-CP-ABE to get $\boldsymbol{k}(\pi s', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r})$. It sets $c'_1 = s'$ and $\boldsymbol{c}'(S, \boldsymbol{b}(\boldsymbol{w}', 1, \boldsymbol{h}'); s', \boldsymbol{s}') := (c'_1, \boldsymbol{k}(\pi s', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r}))$. It implicitly sets $\boldsymbol{s}' = \boldsymbol{r}$.
- Pair$(S, \tilde{\mathbb{A}})$: Pair$(S, \tilde{\mathbb{A}})$ of NM-CP-ABE outputs $\boldsymbol{E}$ such that $\boldsymbol{k}\boldsymbol{E}\boldsymbol{c}^\top = \pi s s'$. The algorithm computes $d'_1 \cdot c'_1 = \alpha s' + \pi s s'$. Finally, the algorithm computes $\alpha s' = d'_1 \cdot c'_1 - \boldsymbol{k}\boldsymbol{E}\boldsymbol{c}^\top$.

A Appendix

A.1 Syntax of Pair Encoding Framework

We briefly introduce Attrapadung's pair encoding framework [3]. In pair encoding, instances for a predicate $R_\kappa : \mathcal{X} \times \mathcal{Y} \rightarrow \{0,1\}$ consist of four deterministic algorithms which are Param, $\mathsf{Enc1}$, $\mathsf{Enc2}$ and Pair.

- $\mathsf{Param}(\kappa) \rightarrow \omega$: It takes as input an index κ and outputs the number of common variables ω of $\boldsymbol{b} = (b_1, ..., b_\omega)$. The common variables are shared with $\mathsf{Enc1}$ and $\mathsf{Enc2}$.
- $\mathsf{Enc1}(x) \rightarrow (\boldsymbol{k} := (k_1, ..., k_{m_1}); m_2)$: It takes as $x \in \mathcal{X}$ and outputs a sequence of polynomials of $\{k_i\}_{i \in [m_1]}$ with coefficient in $\mathbb{Z}_p$ and m_2 which is the number of variables. Every k_i is a linear combination of monomials α, r_k, $b_j r_k$ where $k \in [m_2]$ and $\alpha, r_1, ..., r_{m_2} \in \mathbb{Z}_p$ are variables.
  $\mathsf{Enc2}(y) \rightarrow (\boldsymbol{c} := (c_1, ..., c_{w_1}); w_2)$ It takes as $y \in \mathcal{Y}$ and outputs a sequence of polynomials of $\{c_i\}_{i \in [1, w_1]}$ with coefficient in $\mathbb{Z}_p$ and w_2 which is the number of variables. Every c_i is a linear combination of monomials s, s_k, $b_j s$, $b_j s_k$ where $k \in [w_2]$ and $s, s_1, ..., s_{w_2} \in \mathbb{Z}_p$ are variables.
- $\mathsf{Pair}(x, y) \rightarrow \boldsymbol{E}$ takes as inputs x and y and outputs a reconstruction matrix $\boldsymbol{E}$ such that $\boldsymbol{k}\boldsymbol{E}\boldsymbol{c}^\top = \alpha s$.

The instances of the pair encoding framework satisfy multiple properties, namely *linearity in random variables*, *parameter vanishing* and (*computational* or *perfect*) α *hiding* [3].

References

1. Agrawal, S., Chase, M.: A study of pair encodings: predicate encryption in prime order groups. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016. LNCS, vol. 9563, pp. 259–288. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49099-0_10
2. Agrawal, S., Chase, M.: Simplifying design and analysis of complex predicate encryption schemes. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10210, pp. 627–656. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56620-7_22
3. Attrapadung, N.: Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 557–577. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_31
4. Attrapadung, N.: Dual system encryption framework in prime-order groups. IACR Cryptology ePrint Archive **2015**, 390 (2015)
5. Attrapadung, N.: Dual system encryption framework in prime-order groups via computational pair encodings. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 591–623. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_20
6. Attrapadung, N., Herranz, J., Laguillaumie, F., Libert, B., de Panafieu, E., Ràfols, C.: Attribute-based encryption schemes with constant-size ciphertexts. Theor. Comput. Sci. **422**, 15–38 (2012)

7. Attrapadung, N., Imai, H.: Conjunctive broadcast and attribute-based encryption. In: Shacham, H., Waters, B. (eds.) Pairing 2009. LNCS, vol. 5671, pp. 248–265. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03298-1_16
8. Attrapadung, N., Libert, B.: Functional encryption for inner product: achieving constant-size ciphertexts with adaptive security or support for negation. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 384–402. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_23
9. Attrapadung, N., Yamada, S.: Duality in ABE: converting attribute based encryption for dual predicate and dual policy via computational encodings. In: Nyberg, K. (ed.) CT-RSA 2015. LNCS, vol. 9048, pp. 87–105. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-16715-2_5
10. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D., thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
11. Chen, J., Gay, R., Wee, H.: Improved dual system ABE in prime-order groups via predicate encodings. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 595–624. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_20
12. Chen, J., Wee, H.: Fully, (almost) tightly secure IBE and dual system groups. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 435–460. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_25
13. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78967-3_9
14. Kim, J., Susilo, W., Guo, F., Au, M.H.: Functional encryption for computational hiding in prime order groups via pair encodings. Des. Codes Crypt. **86**(1), 97–120 (2018)
15. Lewko, A.: Tools for simulating features of composite order bilinear groups in the prime order setting. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 318–335. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_20
16. Lewko, A., Sahai, A., Waters, B.: Revocation systems with very small private keys. In: IEEE Symposium on Security and Privacy, pp. 273–285. IEEE Computer Society (2010)
17. Lewko, A.B., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. IACR Cryptology ePrint Arch. **2009**, 482 (2009)
18. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-11799-2_27
19. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_12
20. Liu, Z., Wong, D.S.: Practical ciphertext-policy attribute-based encryption: traitor tracing, revocation, and large universe. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 127–146. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_7

21. Okamoto, T., Takashima, K.: Hierarchical predicate encryption for inner-products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10366-7_13
22. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_11
23. Okamoto, T., Takashima, K.: Fully secure unbounded inner-product and attribute-based encryption. In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 349–366. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34961-4_22
24. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) ACM CCS, pp. 195–203. ACM (2007)
25. Waters, B.: Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03356-8_36
26. Yamada, S., Attrapadung, N., Hanaoka, G., Kunihiro, N.: A framework and compact constructions for non-monotonic attribute-based encryption. In: Krawczyk, H. (ed.) PKC 2014. LNCS, vol. 8383, pp. 275–292. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54631-0_16