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 an advanced encoding framework for predicate encryption (PE) in prime order groups. Our framework captures a wider range of adaptively secure PE schemes such as non-monotonic attribute-based encryption by allowing PE schemes to have more flexible structures. Prior to our work, frameworks featuring adaptively secure PE schemes in prime order groups require strong structural restrictions on the schemes. In those frameworks, exponents of public keys and master secret keys of PE schemes, which are also referred to as common variables, must be linear. In our work, we introduce a modular framework which includes non-linear common variables in PE schemes. First, we formalize non-linear structures which can appear in PE by improving Attrapadung's pair encoding framework (Eurocrypt'14). Then, we provide a generic compiler that features encodings under our framework to PE schemes in prime order groups. Particularly, the security of our compiler is proved by introducing a new technique which decomposes common variables into two types and makes one of them be shared between semi-functional and normal spaces on processes of the dual system encryption to mitigate the linear restriction. As instances of our new framework, we introduce new attribute-based encryption schemes supporting non-monotonic access structures, namely non-monotonic ABE, in prime order groups. We introduce adaptively secure non-monotonic ABE schemes having either short ciphertexts (if KP-ABE) or short keys (if CP-ABE) for the first time. Additionally, we introduce the first non-monotonic ABE schemes supporting both adaptive security and multi-use of attributes property in prime order groups.

Key words: pair encoding, non-monotonic access structure, attribute-based encryption, prime order groups, dual system encryption

1 Introduction

Water's dual system encryption [29] is a widely used proof methodology for adaptively secure PE. After the seminal introduction by Waters, it becomes one of the most popular tools to prove the adaptive security of PE. Subsequently, many PE schemes [21, 18, 7, 28, 15] use the dual system encryption to prove their security. Later, Lewko and Waters [22] introduced a new technique for the dual system encryption. Their novel technique (also referred to as a doubly selective technique in [2]) shows that adaptive security of PE can be achieved *computationally* using selective techniques via the dual system encryption. In detail, prior to their technique, a critical part of the dual system encryption was proved only by information theoretical arguments, but they showed that it can be proved by two selective proofs using the information delivered from the adversary's queries. This significantly extends the usage of the dual system encryption to a wider range of encryption schemes.

Wee [30] and Attrapadung [2] introduced generic modular frameworks which generalize the dual system encryption using encodings. They extract properties that are required by the dual system encryption and formalize them through encoding frameworks. They also introduced generic constructions applicable to those encodings and proved that they are adaptively secure only using the properties defined in the frameworks. Therefore, these frameworks give a new insight to the dual system encryption and make proving adaptive security of an encryption scheme much easier since its security can be simply proved by showing that the corresponding encoding scheme of the encryption scheme satisfies the properties that the frameworks required. In particular, Attrapadung's pair encoding framework suggested in [2] generalizes the doubly selective technique into their framework.

Recently, the dual system encryption has been evolved in prime order groups via encodings [11, 4, 1, 16]. However, their generic constructions for encoding scheme commonly impose a structural restriction on PE in prime order groups. In particular, they require exponents of public keys and master secret keys (as also referred to as common variables) to be linear. This also results that keys and ciphertexts of PE schemes to be linear in those variables. For example, we use $h_1, ..., h_m$ to denote common variables of a pair encoding scheme. The linearity of pair encoding framework requires that public keys and master secret keys to be set $g, g^{h_1}, ..., g^{h_m}$ where g is a group generator and cannot have as parameters group elements of which exponents are not linear in h_i such as $g^{h_1^2}$ or $g^{h_1 h_2}$.

Hence, the usage of encoding frameworks is significantly limited by this structure restriction. Morevoer, the restriction cannot be simply addressed by including non-linear exponents into the encoding framework because (1) it is not clear how the dual system encryption can be used to prove the security of PE having non-linearity in prime order groups. Moreover, it still remains an interesting question (2) whether we can define an encoding framework to capture non-linearity of PE.

1.1 Our Contribution

The contribution of this paper is two-fold.

Improved framework. We introduce a modular framework which is applicable to PE schemes having non-linear common variables in prime order groups. Prior to our works, existing frameworks [11, 4, 1, 16] in prime order groups enforce PE schemes to 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 [2] which is one of most popular encoding frameworks for PE and provide a new compiler that features encodings in our improved framework to an real PE schemes in prime order groups. Unlike the pair encoding framework, we decompose common variables which are exponents of public keys and master secret keys into two types, which are shared common variables $\boldsymbol{w}$ and hidden common variables $\boldsymbol{h}$ and restricted only hidden common variables to be linear. We, then, define public keys (and master secret keys) of PE using monomials $\boldsymbol{b}$ which consists of elements of $\boldsymbol{w}$ and $\boldsymbol{h}$. This refinement flexibly describes even non-linear exponents since there is no structural restriction on $\boldsymbol{w}$ unless it is a monomial. At the same time, we set PE schemes still to be linear in hidden common variables, which we call *relaxed linearity in hidden common variables* so that the dual system methodology can be applied for the security analysis. Secondly, we provide a new generic compiler in prime order groups for our framework and prove its security under simple static assumptions which were introduced by Lewko and Waters [20]. We prove security of our new compiler using *computational* arguments based on the doubly selective technique but we provide an additional refinement of the doubly selective technique to handle non-linearity in PE schemes using both types of common variables. We show that our refinement is still feasible by showing multiple new attribute-based encryption schemes as instances.

Instances. As instances of our new encoding, we introduce two new attribute-based encryption (ABE) schemes supporting a non-monotonic access structure as follows:

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

Table 1. Comparisons of NM-CP-ABE schemes in prime order groups

Scheme	Multi-use of Att.	Security	Assumptions	type	NM-CP-ABE CT	NM-CP-ABE Priv. Key
LSW [19]	Yes	Selective	RO+n-MEBDH	KP	$3n+1$	$2t+t'$
AHLLPR [5]	Yes	Selective	n-DBDHE	KP	4	$(N+1)t$
YAHK [31]	Yes	Selective	q-types	CP	$3t+1$	$4n+2$
	Yes	Selective	q-types	KP	$4n+1$	$3t$
OT [26]	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	$3(N+2)t+9$
Scheme 3	Yes	Adaptive	Static + q-types	CP	$9t+6$	$12n+9$
Scheme 4	Yes	Adaptive	Static + q-types	KP	$12n+12$	$9t+9$

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

- Unbound Non-monotonic CP-ABE (Scheme 3).
- Unbound Non-monotonic KP-ABE (Scheme 4).

We introduce new ABE schemes having short parameters which are either short keys (Scheme 1) or short ciphertexts (Scheme 2). Prior to our work, non-monotonic KP-ABE scheme with short ciphertexts [5] is only selectively secure and there is no scheme supporting short keys. Also, we introduce two new unbounded ciphertext-policy ABE schemes supporting a non-monotonic access policy. The new unbounded schemes are truely unbounded since it supports arbitrary attributes and multi-use of attributes at the same time. Existing ABE schemes supporting non-monotonic access structures are restricted by selective security [31, 19] or one-use of attributes [26] where one-use of attributes means that an attribute does not appear more than once in an access policy.

1.2 Our Technique

Syntax of Pair Encoding Framework [2]. Before we explain our technique, we briefly introduce Attrapadung's pair encoding framework.

In pair encoding, instances for a predicate $R_\kappa : \mathcal{X} \times \mathcal{Y} \to \{0,1\}$ consist of four deterministic algorithms which are Param, $\mathsf{Enc1}$, $\mathsf{Enc2}$ and Pair.

$\mathsf{Param}(\kappa) \to \omega$: 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 $\mathsf{Enc2}$.
$\mathsf{Enc1}(x) \to (\boldsymbol{k} := (k_1, ..., k_{m_1}); m_2)$: 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.
$\mathsf{Enc2}(y) \to (\boldsymbol{c} := (c_1, ..., c_{w_1}); w_2)$ 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.
$\mathsf{Pair}(x, y) \to \boldsymbol{E}$ 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*. Those properties are also required to our new encoding. We discuss them further in Section 4.

Difficulty. There are a few works [16, 4, 11, 1] that feature adaptively secure PE schemes in prime order groups. In particular, the works of Kim et al. [16] and Attrapadung [4] include the doubly selective technique which achieves adaptive security using selective security proofs in their frameworks. All works implicitly or explicitly assume that all parameters of encodings are linear in common variables. Particularly, In [4], the author clearly mention that their framework requires stricter structural restrictions. They defines the scheme satisfying those restrictions as a regular encoding. For example, $h_i h_{i'}$ cannot be used and computed in their framework. The work of Kim et al. [16, 1] also explicitly defines linearity in common variables of keys and ciphertexts as a new property for their security analysis. Also, the techniques suggested in [11, 1] assume that linearity in common variables and use them for their proofs, implicitly using the structural definition of pair encodings. As described in the overview of pair encoding, the pair encoding was not defined only by properties, but also required to have a certain structure which is linear in common variables.

Our Solution. Our solution largely adopts the notion of the pair encoding framework. However, the pair encoding framework cannot properly describe non-linear common variables. Therefore, we improves the syntax of pair encoding. The most significant change in our framework is that we decompose variables used as exponents of public keys and master secret keys into two types *hidden common variables* and *shared common variables* to express non-linearity in PE schemes as follows:

- *Hidden common variables* are variables projected into semi-functional space without being correlated with its original values. Moreover, the semi-functional values of hidden common variables must be hidden before they are projected. These variables required by the dual system encryption technique. To satisfy these requirements, the hidden common variables must be linear. All common variables in existing frameworks [11, 4, 1, 16] are hidden common variables.
- *Shared common variables* are also projected into the semi-functional part but their projected values are exactly same as their original values in normal parts. In other words, these variables are shared both in semi-functional parts and normal parts in semi-functional keys or ciphertexts.

In detail, the exponents of public keys and master secret keys in our encoding framework are composition of those two types 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 shared common variables and hidden common variables, 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 linear operation. 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 $\boldsymbol{h}, \boldsymbol{h'} \in \mathbb{Z}_p^{\omega_2}$

$$\boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}) + \boldsymbol{b}(\boldsymbol{w}, b'_0, \boldsymbol{h'}) = \boldsymbol{b}(\boldsymbol{w}, b_0 + b'_0, \boldsymbol{h} + \boldsymbol{h'}).$$

We call this property *relaxed linearity in hidden common variables.*

Notation Previous notation of pair encoding framework cannot properly describe the linearity of common variables. This deficiency makes us adopt a new variable b_0 in our encoding framework as Kim et al. does in their work [16]. In detail, even if hidden common variables of $\boldsymbol{b}$ are linear form (i.e. the maximum degree of those variables is set to be 1), the relaxed linearity in hidden common variables of $\boldsymbol{b}$ cannot properly be notated if some coordinates of $\boldsymbol{b}$ do not have an element of $\boldsymbol{h}$. Therefore, we use a new variable b_0 to denote the change the values during the addition and place b_0 where an element of $\boldsymbol{h}$ does not appear. Therefore, all coordinates of $\boldsymbol{b}$ must contain either b_0 or h_i and linear in those variables.

Dual system encryption in prime order groups We feature the dual system encryption in the prime order groups using *relaxed linearity in hidden common variables.* In particular, we use the technique of Kim et al. [16], which is a generic compiler that works for pair encoding schemes. Their technique generalizes Lewko and Waters' IBE [20] and utilizes it as a building block of a nested dual system encryption to achieve adaptively secure PE scheme in prime order groups. *Linearity in common variables* which they additionally defined in their work is a core property to prove the security of their compiler in prime order groups. Using the property, the common variables are projected into semi-functional parts and hidden before they are projected. In a high level, the simulator sets a common variable $\boldsymbol{h} = d\boldsymbol{h'} + \boldsymbol{h''}$ where $d \in \mathbb{Z}_p$ is given using a group

		Normal parts	Semi-functional parts
KSGA [16]	**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	$\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})$	$\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); s', \boldsymbol{s}')$

Fig. 1. Comparisons of normal and semi-functional pars in encoding frameworks

generator g such as g^d in the instances of an assumption and $\boldsymbol{h}'$ and $\boldsymbol{h}''$ are values generated by the simulator. This setting is essential to hide the values of $\boldsymbol{h}'$ which are projected and forms common variables in semi-functional parts. However, it is available only if the linear operation in common variables such as addition is possible because the simulator cannot explicitly compute $\boldsymbol{h}$ as the value d is only given as a form of g^d. For example, a normal ciphertext $g^{\boldsymbol{c}(\cdot,(\cdot,\boldsymbol{h}),\cdot)}$ can only be computed by $(g^d)^{\boldsymbol{c}(\cdot,(\cdot,\boldsymbol{h}'),\cdot)} g^{\boldsymbol{c}(\cdot,(\cdot,\boldsymbol{h}''),\cdot)}$ using the linearity in common variables.

In our framework, the exponents of public parameters are more complex monomials, but the simulator still can hide common variable before they are projected into semi-functional parts using the *relaxed linearity in hidden common variables* property. For example, we can define the *relaxed linearity* for $\mathsf{Enc1}$ using *relaxed linearity in hidden common variables* as

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

where x is a predicate; α and α' are values denote master secret; and $\boldsymbol{r}$ are random values for the security anlaysis. Therefore, this property allows the simulator to set a monomial $f_i(\boldsymbol{w})h_j = df_i(\boldsymbol{w})h_j'' + f_i(\boldsymbol{w})h_j'$ similarly to Kim et al.'s compiler and projects $f_i(\boldsymbol{w})h_j'$ into the semi-functional space to form a semi-functional key as shown in Fig. 1.

There exists another compiler potentially work in our encodings from Attrapadung [4]. Their compiler is secure under the matrix DH assumption which can be reduced to the standard *DLIN*. In their security analysis, the common values can be projected without correlating with their original values, but random variables are shared between normal parts and semi-functional parts as we show in Fig. 1. Due to this limitation, they redefined the pair encoding to *regular encoding* with extra structural restrictions. Due to these restrictions, considering non-linearity with the regular encoding is quite complex and our motivation is easing the structural restrictions of pair encoding. Because Kim et al.'s technique provides more flexible structure to PE and their security is also secure under static and simple assumptions, we utilize their compiler as a backbone of our compiler.

Refined computational α hiding In our setting, the values of $\boldsymbol{w}$ are not hidden. Those values are projected into semi-functional parts by $f_i(\boldsymbol{w})$, but the projected values are identical with their original values as shown in Fig. 1. Sharing $\boldsymbol{w}$ is not typical in the dual system encryption because it means $\boldsymbol{w}$ must be defined and fixed when a system sets up, which is not required in the dual system encryption.

We address this problem by redefining computational α hiding property of pair encoding framework. We use two oracles which are indistinguishable to each other to simulate the refined computational hiding property. In our setting, the oracles output shared common variables $\boldsymbol{w}$ by include $g^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}$ in initial instances so that the simulator create public keys and normal parts of private keys using $\boldsymbol{w}$. It is worth noting that the oracles in the other techniques [16, 4] only output a group generator g in an initial instance. Fig. 2 shows that the difference of our oracles.

One of the difficulties to construct these oracles is proving our refined computational α hiding property. In the existing pair encoding schemes, the computational α hiding is usually proved via the doubly selective technique. Because the initial instance does not include any public parameters, the oracles can select public parameters after they see the target predicate of the challenge ciphertext or the challenge key . However, this benefit is not valid in ours because our oracles must output shared common variables before they see the target predicate.

	Oracles of [16, 4, 22]	Our Oracles
Initial Instance:	g	$\boxed{g^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}}$
CT Instance:	$g^{\boldsymbol{c}(y,\boldsymbol{h}';s',\boldsymbol{s}')}$	$g^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s',\boldsymbol{s}')}$
Key Instance*:	$g^{\boldsymbol{k}(\alpha',x,\boldsymbol{h}';\boldsymbol{r}')}$	$g^{\boldsymbol{k}(\alpha',x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}')}$

* : α' is either a random value or 0

Fig. 2. Our Refined Computational α-hiding

However, we observed that, even in selective security proofs, some common variables can be set without using any information of the target predicate or some existing selective security proofs can be easily modified to set a part of common variables without knowing the target predicate. This allows us to use the variables independently created from the target predicate as shared common variables and build the oracles because selecting which common variables to be hidden or shared is quite flexible in our framework. This implies that we still can prove the refined computational α hiding property based on existing selective proofs as the other existing pair encoding schemes do. We show that achieving those oracles is feasible providing new instances such as attribute-based encryption schemes supporting non-monotonic access structures .

2 Related Work

Conjunctive schemes with ABE with monotonic access structures and identity based revocation systems were introduced for a revocation [6, 23] to fill the gap between practice and theory when a practical ABE scheme with non-monotonic access structures was absence. In those schemes, only a special attribute such as identity can be used to revoke users and the attribute for the revocation cannot be reused in an access policy. Inner product encryption [14, 24, 7, 25] naturally achieves a non-monotonic access structure using polynomials. However, expressing a Boolean formula using inner product is less efficient than ABE schemes. A technique to feature encryption schemes in composite order groups into prime order groups were introduced by Lewko [17] using Dual Pairing Vector Spaces (DPVS) [24, 25]. However, their conversion technique is not generic and the efficiency of a converted scheme linearly increases in the size of vector it uses. Dual System Groups (DSG) [12] were recently introduced. Chen and Wee showed that DSG can be utilized to construct a wide range of encryption schemes in prime order groups. Many generic constructions [11, 1, 4] for encoding schemes in prime order groups utilize DSG except Kim et al.'s work [16]. Instead of using DSG, Kim et al.[16] generalized the Lewko and Waters' IBE [20] to construct the general construction.

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 $e : G_1 \times G_2 \to G_T$ 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_1 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-monotonic access Structure

Definition 1 *(Access Structure) [9] 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 monotonic 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)) [9] 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-monotonic access structures For a non-monotonic access structure, we adopt a technique from Ostrovsky, Sahai and Waters [27]. They assume a family of linear secret sharing schemes $\{\Pi_{\mathbb{A}}\}_{\mathbb{A}\in\mathcal{A}}$ for a set of monotonic 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 $N(\tilde{S}) = \tilde{S} \cup \{x' | x \in \tilde{P} \backslash \tilde{S}\}$. For each access structure $\mathbb{A} \in \mathcal{A}$ over a set of parties $\mathcal{P}$, a non-monotonic 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-monotonic 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-monotonic 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 [16, 20]. 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 [2].

Definition of Predicate Encryption [2]. A PE for a function R_κ consists of Setup, Encrypt, KeyGen and 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. It outputs a public parameter PK and a master secret key MSK.
$\mathsf{Encrypt}(x, M, PK) \to CT$: The algorithm takes in a predicate $x \in \mathcal{X}$, a public parameter PK and a plaintext M. It outputs a ciphertext CT.
$\mathsf{KeyGen}(y, MSK, PK) \to SK$: The algorithm takes in a predicate $y \in \mathcal{Y}$, 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.

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 Functional Encryption [2]. A functional encryption for a 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 an adversary $\mathcal{A}$'s advantage as $Adv_{\mathcal{A}}^{PE}(\lambda) := |\Pr[b = b'] - 1/2|$.

4 Our Encoding Framework

In this section, 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 requires non-linear parameters.

4.1 Syntax

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

$\mathsf{Param}(\kappa) \to (\boldsymbol{b} := (b_1, b_2, ..., b_\omega); \omega_1, \omega_2, \omega)$: 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 shared common variables and $\boldsymbol{h} = (h_1, ..., h_{\omega_2})$ denote the hidden common variables. b_0 is a variable for the linearity***.
$\mathsf{Enc}_1(x \in \mathcal{X}) \to (\boldsymbol{k} := (k_1, k_2, ..., k_{m_1}); m_2)$: It takes as inputs a predicate 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 $i \in [m_1]$,

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

*** b_0 is used for the security analysis. It is fixed as 1 in an encoding scheme (and its actual construction). It has other values only in security proofs where $\delta_i, \delta_{i,j}, \delta_{i,j,k} \in \mathbb{Z}_p$ are constants which define k_i.

$\mathsf{Enc}_2(y \in \mathcal{Y}) \to (\boldsymbol{c} := (c_1, c_2, ..., c_{\tilde{m}_1}); \tilde{m}_2)$: It takes as inputs a predicate 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 $sb_0, s_ib_0, sb_j, s_ib_j$ in variables $s, s_1, ..., s_{\tilde{m}_2}$ and $b_0, b_1, ..., b_\omega$. In more detail, for $i \in [\tilde{m}_1]$,

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

where $\phi_i, \phi_{i,j}, \phi_{i,j,k} \in \mathbb{Z}_p$ are constants which define c_i.

$\mathsf{Pair}(x, y) \to \boldsymbol{E}$: 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 $\boldsymbol{kEc}^\top = \alpha s$ where $\boldsymbol{kEc}^\top = \sum_{i\in[m_1],j\in[\tilde{m}_1]} E_{i,j}k_ic_j$.

4.2 Properties

We describe properties that our encodings have.

Property 1. (Relaxed 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 $\boldsymbol{0} \in \mathbb{Z}_p^{2k+1}$ which makes following two distributions are statistically identical:

$$\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, b_0, \boldsymbol{h}); \boldsymbol{0}) \text{ and } \boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, b_0', \boldsymbol{h}'); \boldsymbol{0}).$$

Property 4. (Computational α hiding) We let $g_1 \in G_1$ and $g_2 \in G_2$ be selected randomly. For all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ such that $R_\kappa(x, y) = 0$, the following two distributions are *computationally* indistinguishable:

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

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

where $\alpha, s \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{w} \xleftarrow{R} \mathbb{Z}_p^{\omega_1}$, $\boldsymbol{h} \xleftarrow{R} \mathbb{Z}_p^{\omega_2}, \boldsymbol{r} \xleftarrow{R} \mathbb{Z}_p^{w_2}$ and $\boldsymbol{s} \xleftarrow{R} \mathbb{Z}_p^{m_2}$.

To prove the *computational α hiding*, we define oracles $\mathcal{O}_\beta^{Cos}$ and $\mathcal{O}_\beta^{Sel}$ where $\beta = \{0, 1\}$. $\mathcal{O}_\beta^{Cos}$ and $\mathcal{O}_\beta^{Sel}$ simulates computational α hiding for $\mathsf{Phase\ I}$ (Co-selective Security) and $\mathsf{Phase\ II}$ (Selective Security) of the adaptive security model, respectively. The responses of oracles are defined following:

- Initial instance: $\{g_1^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}\}$
- k-type response: $g_2^{\boldsymbol{k}(\beta\cdot\alpha,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})}$

- c-type response: $g_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}$

Oracles for co-selective Security$\mathcal{O}_\beta^{Cos}$: When the oracle receives an initial query before it receives any other query, it outputs the initial instance After the initial instance, the oracle only can respond to the k-type query. When the oracle receives the k-type query for a predicate x, it sends the k-type response. After it responds, it can respond to the c-type query for a description y if $R_\kappa(x, y) = 0$. When the oracle receives the c-type query for y, it outputs the c-type response.

Oracles for selective Security$\mathcal{O}_\beta^{Sel}$: This oracle is identical to $\mathcal{O}_\beta^{Cos}$ except the order of the responses. This oracle first outputs the initial instance. Then, it outputs the k-type response before the c-type response.

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**. We use subscripts to denote where group elements belong (e.g. $g_1 \in G_1$, $g_2 \in G_2$).

- **Setup**$(1^\lambda, \kappa) \to \langle PK, MSK\rangle$. The setup algorithm randomly chooses bilinear groups $\mathbb{G} = (p, G_1, G_2, G_T)$ of prime order $p > 2^\lambda$. It takes group generators $g_1 \xleftarrow{R} G_1, g_2 \xleftarrow{R} G_2$ from $\mathbb{G}$. 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**$(x, MSK) \to SK$. 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 $\boldsymbol{z} \in \mathbb{Z}_p^{m_1}$ where $m_1 = |\boldsymbol{k}|$. It parses α from MSK and outputs $SK := (\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3)$ following:
$$\boldsymbol{D}_1 = g_2^{\boldsymbol{k}(\alpha,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})} v_2^{\boldsymbol{z}},\ \boldsymbol{D}_2 = u_2^{\boldsymbol{z}},\ \boldsymbol{D}_3 = f_2^{-\boldsymbol{z}}.$$
- **Encrypt**$(M, y, PK) \to CT$. 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 $CT := (C_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$ following:
$$\boldsymbol{C}_1 = g_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}, \boldsymbol{C}_2 = (g_1^a)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}, \boldsymbol{C}_3 = (g_1^\tau)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}.$$
- **Decrypt**$(x, y, SK, CT) \to M$. It takes as inputs SK for $x \in \mathcal{X}$ and CT for $y \in \mathcal{Y}$. It runs $E \leftarrow \mathsf{Pair}(x, y)$ 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 $\boldsymbol{E}$ such that $\boldsymbol{cEk}^\top = \alpha s$ because $b_0 = 1$. Therefore, we can compute followings:
$$A_1 = e(\boldsymbol{C}_1^{\boldsymbol{E}^\top}, \boldsymbol{D}_1) = e(g_1, g_2)^{\boldsymbol{cE}^\top\boldsymbol{k}^\top} e(g_1, v_2)^{\boldsymbol{cE}^\top\boldsymbol{z}^\top} = e(g_1, g_2)^{\alpha s} e(g_1, v_2)^{\boldsymbol{cE}^\top\boldsymbol{z}^\top}$$
$$A_2 = e(\boldsymbol{C}_2^{\boldsymbol{E}^\top}, \boldsymbol{D}_2) = e(g_1, u_2)^{a\cdot\boldsymbol{cE}^\top\boldsymbol{z}^\top},\ A_3 = e(\boldsymbol{C}_3^{\boldsymbol{E}^\top}, \boldsymbol{D}_3) = e(g_1, f_2)^{-\tau\cdot\boldsymbol{cE}^\top\boldsymbol{z}^\top}$$

It should be noted that $\tau = y_v + ay_u$ 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 $\mathcal{B}_1$, $\mathcal{B}_2$, $\mathcal{B}_3$, $\mathcal{B}_4$ and $\mathcal{B}_5$, whose running times are the same as $\mathcal{A}$ such that, for any λ,*

$$Adv_{\mathcal{A}}^{FE(P)}(\lambda) \leq w_t \cdot Adv_{\mathcal{B}_1}^{LW1}(\lambda) + 2 \cdot m_t \cdot Adv_{\mathcal{B}_2}^{LW2}(\lambda) + Adv_{\mathcal{B}_3}^{LW3}(\lambda)$$

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

where q_1 and q_2 are the numbers of key queries in phases I and II, respectively, and m_t is the total number of random variables used to simulate all private keys and w_t is the number of random variables used in the challenge ciphertext.

5 Security Analysis

We define the semi-functional (SF) algorithms to security proofs. To create various types of keys and the challenge ciphertext, the simulator first randomly selects $\boldsymbol{h}' \in \mathbb{Z}_p^{\omega_2}$ which is shared in semi-functional algorithms.

SFKeyGen$(x, MSK, \boldsymbol{h}', j, \alpha') \rightarrow SK$: The algorithm takes as inputs the master secret key MSK, $x \in \mathcal{X}$ and $j \in [m_2]$. 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 $SK := \langle \boldsymbol{D}_1', \boldsymbol{D}_2', \boldsymbol{D}_3' \rangle$ following:

$$\boldsymbol{D}_1' = \boldsymbol{D}_1 \cdot f_2^{-a\boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{\boldsymbol{r}}_j)}, \boldsymbol{D}_2' = \boldsymbol{D}_2 \cdot f_2^{-\tau\boldsymbol{k}(\alpha', x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{\boldsymbol{r}}_j)}, \boldsymbol{D}_3' = \boldsymbol{D}_3$$

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 particular, in SF keys, $\tilde{\boldsymbol{r}}_0$ equals the zero vector $\boldsymbol{0}$ by the definition. Due to the parameter vanishing property, it does not require $\boldsymbol{h}'$ as the inputs and we can rewrite 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 $j \in [w_2+1]$. 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 $j = 0$, 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 $j > 0$, it selects a random value $\tilde{s} \xleftarrow{R} \mathbb{Z}_p$ and a random vector $\tilde{\boldsymbol{s}}_j \xleftarrow{R} \mathbb{Z}_p^{w_2}$ where the first j 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 $j = \tilde{m}_2$.

We describe the security games that we use for the security proof in Table 2. In the proof, we will show that all games in Table 2 are indistinguishable.

The most critical part among them is the invariance between games G_{k,m_2}^N and G_{k,m_2}^T where m_2 is the number of variables in the k^{th} key. There are two cases based on the value of k, either

Table 2. Games for Security Analysis

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

$k \leq q_1$ or $k > q_1$ where the q_1 is the number of key queries that the adversary requests before it requests the challenge ciphertext. We provide the proofs of those two cases in Lemmas 3 and 4.

Lemma 1.1. Suppose there exists a PPT $\mathcal{A}$ who can distinguish G_{Real} and $\mathsf{G}_{0,0}$ with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ which breaks $LW1$ with the advantage ϵ using $\mathcal{A}$.

Proof: In this proof, given instance from $LW1$

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

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

Setup: For the predicate family with an index κ, $\mathcal{B}$ run $\mathsf{Param}(\kappa)$ to generate $\boldsymbol{b}, \omega_1, \omega_2$ and ω. It, then, randomly selects $\alpha, y_g, y_u \in \mathbb{Z}_p$, $\boldsymbol{w} \in \mathbb{Z}_p^{w_1}$, $\boldsymbol{h}', \boldsymbol{h}'' \in \mathbb{Z}_p^{w_2}$ and it sets

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

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

The values of $\boldsymbol{h}$ are set implicitly by $g_1^{\boldsymbol{h}} = f_1^{(c^2 \boldsymbol{h}' + \boldsymbol{h}'')}$ since the simulator does not know the value of c^2. Also, it implies that $\tau = c + a y_u$, $y_v = c$ and $y_f^{-1} = c^2 + y_g$. It publishes public parameters as follows:

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

$$g_1^\tau = f_1^{c^3} (f_1^{ac^2})^{y_u} (f_1^c)^{y_g} (f_1^a)^{y_u y_g}, g_1^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = (f_1^{c^2})^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')} f_1^{\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'')},$$

$$g_1^{a \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = (f_1^{ac^2})^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')} (f_1^a)^{\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'')},$$

$$g_1^{\tau \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = (f_1^{c^3})^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')} (f_1^c)^{\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'')} (f_1^{ac^2})^{y_u \cdot \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}')} (f_1^a)^{y_u \cdot \boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'')}\}.$$

Because $f_2^{c^2}$ is not given, $\mathcal{B}$ cannot explicitly generate MSK. But, we show that $\mathcal{B}$ still properly-generates a privates key for $x \in \mathcal{X}$ only using f_2^c in $\mathsf{Phase\ I/II}$.

$\mathsf{Phase\ I/II}$: To generate a normal private key for $x \in \mathcal{X}$, $\mathcal{B}$ randomly selects $\boldsymbol{z}' \in \mathbb{Z}_p^{m_k}$ and $\boldsymbol{r} \in \mathcal{R}_r$ where $m_k = |\boldsymbol{k}(\alpha, x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \boldsymbol{r})|$. It, then, implicitly sets

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

$\boldsymbol{z}$ is randomly distributed due to $\boldsymbol{z}'$. $\mathcal{B}$ can create normal key as follows:

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

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

$\boldsymbol{K}_0, \boldsymbol{K}_1$ and $\boldsymbol{K}_2$ can be calculated since f_2^c is given. Also, they are properly distributed since

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

The equalities (1) and (2) hold because of *relaxed linearity in hidden common variables* property.

Challenge: When the adversary asks the challenge ciphertext with messages M_0 and M_1. $\mathcal{B}$ randomly chooses $\beta \in \{0,1\}$ and randomly selects $s_1, ..., s_{\tilde{m}_2} \in \mathbb{Z}_p$. It sets $s = d$ and $\boldsymbol{s} = (s_1, ..., s_{\tilde{m}_2})$. The value of d has never been used. Therefore, setting $s = d$ is hidden to the adversary. It creates the challenge ciphertext as

$$\begin{aligned}
C =& M_\beta \cdot e(f_1^{c^3 s}, f_2^c)^\alpha e(f_1^{c^2 s}, f_2)^{2\alpha \cdot y_g} e(f_1^s, f_2)^{\alpha \cdot y_g^2}, \\
\boldsymbol{C}_0 =& (f_1^{c^2})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); 0, \boldsymbol{s})} f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, y_g, \boldsymbol{h}''); 0, \boldsymbol{s})} (f_1^{c^2 d})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); 1, \boldsymbol{0})} \\
& \cdot (f_1^d)^{\boldsymbol{c}(y, (\boldsymbol{b}(\boldsymbol{w}, y_g, \boldsymbol{h}''); 1, \boldsymbol{0})} \\
\boldsymbol{C}_1 =& (f_1^{ac^2})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); 0, \boldsymbol{s})} T^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); 1, \boldsymbol{0})} (f_1^a)^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, y_g, \boldsymbol{h}''); 0, \boldsymbol{s})} \\
& \cdot (f_1^{ad})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, y_g, \boldsymbol{h}''); 1, \boldsymbol{0})}
\end{aligned}$$

and sets $\boldsymbol{C}_2 = \boldsymbol{C}_0^c \cdot \boldsymbol{C}_1^{y_u}$. $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ can be calculated by the given instances. Also, $\boldsymbol{C}_2$ 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 $T = f_1^{ac^2 d}$, the challenge ciphertext is normal and generated using KeyGen. Hence, the algorithm has properly simulated Game_{Real}. Otherwise, if T is a random value, we use $T = f_1^{ac^2 d} f_1^{\tilde{s}}$ to denote it. Then, $f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); \tilde{s}, \boldsymbol{0})}$ appears in the challenge ciphertext, which is identical with the output of $\mathsf{SFEncrypt}(M, y, PK, \boldsymbol{h}', 0)$ and $\mathsf{Game}_{0,0}$ has been simulated.

□

Lemma 1.2. Suppose there exists a PPT $\mathcal{A}$ who can distinguish $\mathsf{G}_{0,i-1}$ and $\mathsf{G}_{0,i}$ for $i \in [\tilde{m}_2]$ with non-negligible advantage ϵ where w_2 is the number of random variables that the challange ciphertext has. Then, we can build an algorithm $\mathcal{B}$ which breaks $LW1$ with the advantage ϵ using $\mathcal{A}$.

Proof: This lemma is almost identical with Lemma 1.1. except Challenge. $\mathcal{B}$ simulates Challenge as follows:

Challenge: When the adversary asks the challenge ciphertext with messages M_0 and M_1. $\mathcal{B}$ randomly chooses $\beta \in \{0,1\}$ and randomly selects $s, s_1, ..., s_{i-1}, s_{i+1}, ..., s_{\tilde{m}_2}, s', s'_1, ..., s'_{i-1} \in \mathbb{Z}_p$. It sets $\boldsymbol{s} = (s_1, ..., s_{i-1}, d, s_{i+1}, ..., s_{\tilde{m}_2})$ and $\boldsymbol{s}'_{i-1} = (s'_1, ..., s'_{i-1}, 0, ..., 0)$. The value of d has never been used. Therefore, $\boldsymbol{s}$ is uniformly random to the adversary. It calculates the challenge ciphertext as

$$
\begin{aligned}
C =& M_\beta \cdot e(f_1^{c^3 s}, f_2^c)^\alpha e(f_1^{c^2 s}, f_2)^{2\alpha\cdot y_g} e(f_1^s, f_2)^{\alpha\cdot y_g^2},\\
\boldsymbol{C}_0 =& (f_1^{c^2})^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s,\boldsymbol{s}-d\cdot\mathbf{1}_i)} (f_1^{c^2 d})^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\mathbf{1}_i)} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');s,\boldsymbol{s}-d\cdot\mathbf{1}_i)},\\
&\cdot (f_1^d)^{\mathbf{c}(y,(\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');0,\mathbf{1}_i)}\\
\boldsymbol{C}_1 =& (f_1^{ac^2})^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s,\boldsymbol{s}-d\cdot\mathbf{1}_i)} T^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\mathbf{1}_i)} (f_1^a)^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');s,\boldsymbol{s}-d\cdot\mathbf{1}_i)}\\
&\cdot (f_1^{ad})^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');0,\mathbf{1}_i)} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'),s',\boldsymbol{s}'_{i-1})}
\end{aligned}
$$

and sets $\boldsymbol{C}_2 = \boldsymbol{C}_0^c \cdot \boldsymbol{C}_1^{y_u}$ where $\mathbf{1}_i$ is a vector of which only the ith coordinate is 1 and all other coordinates are 0. It should be noted that $\boldsymbol{s} - d\cdot\mathbf{1}_i$ is equal to $(s_1, ..., s_{i-1}, 0, s_{i+1}, ..., s_{\tilde{m}_2})$. Hence, it does not have d as a coordinate. Therefore, $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ can be calculated by the given instances. Also, $\boldsymbol{C}_2$ 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 $T = f_1^{ac^2 d}$, the challenge ciphertext is the normal challenge ciphertext since

$$
\begin{aligned}
\boldsymbol{C}_1 =& (f_1^{ac^2})^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');s,\boldsymbol{s}-d\cdot\mathbf{1}_i)} f_1^{ac^2 d\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\mathbf{1}_i)} (f_1^a)^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');s,\boldsymbol{s}-d\cdot\mathbf{1}_i)}\\
&\cdot (f_1^{ad})^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');0,\mathbf{1}_i)} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'),s',\boldsymbol{s}'_{i-1})}\\
=& f_1^{a\cdot\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},c^2,c^2\boldsymbol{h}');s,\boldsymbol{s})} f_1^{a\cdot\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},y_g,\boldsymbol{h}'');s,\boldsymbol{s})} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'),s',\boldsymbol{s}'_{i-1})}\\
=& f_1^{a\cdot\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},c^2+y_g,c^2\boldsymbol{h}'+\boldsymbol{h}'');s,\boldsymbol{s})} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'),s',\boldsymbol{s}'_{i-1})}\\
=& g_1^{a\cdot\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}),s,\boldsymbol{s})} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'),s',\boldsymbol{s}'_{i-1})} \qquad (3)
\end{aligned}
$$

- If T is random value and we let $T = f_1^{ac^2 d} f_1^\gamma$,

$$
T^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\mathbf{1}_i)} = f_1^{ac^2 d\cdot\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\mathbf{1}_i)} f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\gamma\cdot\mathbf{1}_i)}
$$

Therefore, $f_1^{\mathbf{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');0,\gamma\cdot\mathbf{1}_i)}$ is multiplied to (3). It means that

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

where $\boldsymbol{s}'_i = (s'_1, ..., s'_{i-1}, \gamma, 0, ...0)$.

Therefore, if $T = f_1^{ac^2 d}$, the challenge ciphertext is generated using $\mathsf{SFEncrypt}\ (M, y, PK, \boldsymbol{h}', i-1)$ and $\mathsf{Game}_{0,i-1}$ has been simulated. Otherwise, the challenge ciphertext is generated using $\mathsf{SFEncrypt}(M, y, PK, \boldsymbol{h}', i)$ and $\mathsf{Game}_{0,i}$ has been simulated. □

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 $\mathcal{A}$.

Proof: Using the given instance $\{f_1, f_1^d, f_1^{d^2}, f_1^{tw}, f_1^{dtw}, f_1^{d^2 t} \in G_1, f_2, f_2^c, f_2^d, f_2^w, T \in G_2\}$, $\mathcal{B}$ will simulate either $\mathsf{Game}_{k,j-1}^N$ or $\mathsf{Game}_{k,j}^N$ 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 publishes a public key as

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

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

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

Then, it sets

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

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

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 $\mathbb{Z}_p^{m_k}$ where $m_k = |\boldsymbol{k}|$ 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**$(x, MSK, \boldsymbol{h}', j-1, 0)$ since

$$\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^{wa})^{-1} f_2^{y'_v})^{\boldsymbol{z}'} (f^{cw})^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\cdot\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})}\\
=& 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})} && (4)\\
=& 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))}\\
&\cdot f_2^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})} && (5)\\
=& 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))}\\
&\cdot f_2^{-a\boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\boldsymbol{r}'_{j-1})} && (6)\\
=& 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}$$

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

This implicitly sets $\boldsymbol{r} = \boldsymbol{r}'' - c \cdot \mathbf{1}_j$ and $\boldsymbol{z} = \boldsymbol{z}' + \boldsymbol{k}(0, x, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}'); c \cdot \mathbf{1}_j)$. The second equality (4) in above equation holds by the linearity over random values. The third equality (5) holds because of the definition of $\boldsymbol{r}$ $(= \boldsymbol{r}'' - c \cdot \mathbf{1}_j)$ and linearity over random values. The equality (6) holds due to relaxed linearity over 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 $\mathbf{SFKeyGen}(x, MSK, \boldsymbol{h}', j, 0)$ since this implicitly sets $\boldsymbol{r}'_j = \boldsymbol{r}'_{j-1} + \gamma \cdot \mathbf{1}_j$. 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 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 $\boldsymbol{s}'', \tilde{\boldsymbol{s}} \in \mathcal{R}_s$. 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'', $\tilde{s}$, $\tilde{\boldsymbol{s}}$ and $\boldsymbol{s}''$, 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

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

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

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

The challenge ciphertext is also properly distributed because

$$\begin{aligned}
\boldsymbol{C}_0 &= (f_1^{dwt})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); \tilde{s}, \tilde{\boldsymbol{s}})} (f_1^d)^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); s'', \boldsymbol{s}'')} (f_1^{wt})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}''); \tilde{s}, \tilde{\boldsymbol{s}})} \\
&\quad \cdot f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}''); s'', \boldsymbol{s}'')} \\
&= (f_1^{wt})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},d,d\boldsymbol{h}'); \tilde{s}, \tilde{\boldsymbol{s}})} (f_1)^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},d,d\boldsymbol{h}'); s'', \boldsymbol{s}'')} (f_1^{wt})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}''); \tilde{s}, \tilde{\boldsymbol{s}})} \\
&\quad \cdot f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}''); s'', \boldsymbol{s}'')} && (7) \\
&= f_1^{\boldsymbol{c}(y, (d, d\boldsymbol{h}'); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} f_1^{\boldsymbol{c}(y, (0, \boldsymbol{h}''); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} && (8) \\
&= f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, d, d\boldsymbol{h}'+\boldsymbol{h}''); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} && (9) \\
&= g_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); s, \boldsymbol{s})}
\end{aligned}$$

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

$$\begin{aligned}
\boldsymbol{C}_2 &= (f_1^{d^2})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); s'', \boldsymbol{s}'')} (f_1^{dwt})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, y'_v, \boldsymbol{h}''+y'_v\boldsymbol{h}'); \tilde{s}, \tilde{\boldsymbol{s}})} \\
&\quad \cdot (f_1^d)^{c(y, \boldsymbol{b}(\boldsymbol{w}, y'_v, \boldsymbol{h}''+y'_v\boldsymbol{h}'); s'', \boldsymbol{s}'')} (f_1^{wt})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,y'_v\boldsymbol{h}''); \tilde{s}, \tilde{\boldsymbol{s}})} f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,y'_v\boldsymbol{h}''); s'', \boldsymbol{s}'')} \\
&= (f_1^{d^2})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} (f_1^{d^2})^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); -wt\tilde{s}, -wt\tilde{\boldsymbol{s}})} \\
&\quad \cdot (f_1^d)^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, y'_v, \boldsymbol{h}''+y'_v\boldsymbol{h}'); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},0,y'_v\boldsymbol{h}''); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} && (10) \\
&= f_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w}, (d+y'_v)d, d(d+y'_v)\boldsymbol{h}'+(d+y'_v)\boldsymbol{h}''); wt\tilde{s}+s'', wt\tilde{\boldsymbol{s}}+\boldsymbol{s}'')} \\
&\quad \cdot (f_1^w)^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); -d^2t\tilde{s}, -d^2t\tilde{\boldsymbol{s}})} && (11) \\
&= g_1^{\tau \cdot \boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); s, \boldsymbol{s})} u_1^{\boldsymbol{c}(y, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}'); s', \boldsymbol{s}')}.
\end{aligned}$$

The equalities of (7) and (9) hold by *relaxed linearity over common variables*. Also, those of (8) and (10) hold by linearity over random values. The equalities of (11) holds due to both linearity over random values and relaxed linearity over common variables. 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 $s = wt\tilde{s} + s''$, $\boldsymbol{s} = wt\tilde{s} + \boldsymbol{s}''$ since their values are not revealed in those values (due to s'' and $\boldsymbol{s}''$).

□

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 $k < q_1$. Then, we can build an algorithm $\mathcal{B}$ who can distinguish between $\mathcal{O}^{Cos}_0$ and $\mathcal{O}^{Cos}_1$ with ϵ using $\mathcal{A}$.

Proof: Given a PPT adversary $\mathcal{A}$ who can distinguish G^N_{k,m_2} and G^T_{k,m_2} with a non-negligible advantage for $k < q_1$, we will build an algorithm $\mathcal{B}$ to distinguish between $\mathcal{O}^{Cos}_0$ and $\mathcal{O}^{Cos}_1$. $\mathcal{B}$ works with either $\mathcal{O}^{Cos}_0$ or $\mathcal{O}^{Cos}_1$. Depending on the oracle $\mathcal{B}$ works with, it will simulate G^N_{k,m_2} and G^T_{k,m_2} with $\mathcal{A}$.

Setup: When $\mathcal{A}$ requests the PK, $\mathcal{B}$ requests the initial instance the oracle, it works with then receives $\{\omega_1, \omega_2, \omega, g_1^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1})}\}$. Then, it randomly creates $\alpha, a, y_f, y_u, y_v \in \mathbb{Z}_p$ and $\boldsymbol{h} \in \mathbb{Z}_p^{\omega_2}$. It sets $u_1 = g_1^{y_f y_u}$, $f_1 = g_1^{y_f}$ and $\tau = y_v + a \cdot y_u$, and publishes the public parameters:

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

It sets MSK as $\{\alpha, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, f_2 = g_2^{y_f}, u_2 = g_2^{y_f y_u}, v_2 = g_2^{y_f y_v}\}$. It should be noted that all elements of the public key and the master secret key can be computed since $\mathcal{B}$ knows all exponents unless they do not contain $\boldsymbol{w}$ and $\boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h})$ is linear in $\boldsymbol{h}$.

$\mathsf{Phase\ I}$: For the first $k-1$ keys for S_i, $\mathcal{B}$ generates a normal key $(\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3)$ using **KeyGen**. It randomly creates $\alpha'_i \in \mathbb{Z}_p$ and sets

$$\boldsymbol{D}'_1 = \boldsymbol{D}_1 \cdot f_2^{-a\boldsymbol{k}(\alpha'_i, x_i, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0});\boldsymbol{0})}, \boldsymbol{D}'_2 = \boldsymbol{D}_2 \cdot f_2^{\boldsymbol{k}(\alpha'_i, x_i, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0});\boldsymbol{0})}, \boldsymbol{D}'_3 = \boldsymbol{D}_3$$

$f_2^{-a\boldsymbol{k}(\alpha'_i, x_i, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0});\boldsymbol{0})}$ and $f_2^{\boldsymbol{k}(\alpha'_i, x_i, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0});\boldsymbol{0})}$ can be computed since it knows α'_i. It sends a semi-functional key $(\boldsymbol{D}'_1, \boldsymbol{D}'_2, \boldsymbol{D}'_3)$. For the key queries after the k^{th} key including keys in $\mathsf{Phase\ II}$, $\mathcal{B}$ sends normal keys using **KeyGen**. $\mathcal{B}$ can compute **KeyGen** since it knows all PK and MSK.

To create the k^{th} key for x_k, it sends k-type query for x_k to the oracle it works with. After receiving $g_2^{\boldsymbol{k}(\beta\cdot\alpha', S, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{\boldsymbol{r}})}$ from the oracle, $\mathcal{B}$ generates a normal key for $(\boldsymbol{D}_1, \boldsymbol{D}_2, \boldsymbol{D}_3)$ using **KeyGen** and sets

$$\boldsymbol{D}'_1 = \boldsymbol{D}_1 \cdot (g_2^{\boldsymbol{k}(\beta\cdot\alpha', S, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{\boldsymbol{r}})})^{-a\cdot y_f}, \boldsymbol{D}'_2 = \boldsymbol{D}_2 \cdot (g_2^{\boldsymbol{k}(\beta\cdot\alpha', S, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{\boldsymbol{r}})})^{y_f}, \boldsymbol{D}'_3 = \boldsymbol{D}_3.$$

If the simulator works with $\mathcal{O}^{Cos}_0$, $\mathcal{B}$ simulates $\textbf{SFKeyGen}(x_k, MSK, \boldsymbol{h}', m_2, 0)$. Therefore, this properly simulates G^N_{k,m_2}. If the simulator works with $\mathcal{O}^{Cos}_1$, $\mathcal{B}$ simulates $\textbf{SFKeyGen}(x_k, MSK, \boldsymbol{h}', m_2, \alpha')$. This properly simulates G^T_{k,m_2}.

$\mathsf{Challenge}$: The adversary requests the challenge ciphertext for a message M and a predicate y. $\mathcal{B}$ generates the normal ciphertext $(C_0, \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3)$ by running **Encrypt** using the public key. It sends the c-type query for y to the oracle which it works with and receives $g_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{s},\tilde{\boldsymbol{s}})}$. $\mathcal{B}$ sets $C'_0 = C_0, \boldsymbol{C}'_1 = \boldsymbol{C}_1$ and

$$\boldsymbol{C}'_2 = \boldsymbol{C}_2 \cdot (g_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{s},\tilde{\boldsymbol{s}})})^{y_f}, \ \boldsymbol{C}'_3 = \boldsymbol{C}_3 \cdot (g_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}');\tilde{s},\tilde{\boldsymbol{s}})})^{y_f\cdot y_u}.$$

It outputs the semi-functional challenge ciphertext $CT' = (C'_0, \boldsymbol{C}'_1, \boldsymbol{C}'_2, \boldsymbol{C}'_3)$.

□

Lemma 4. 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 and the k^{th} key query is given after the challenge ciphertext query ($k > q_1$). Then, we can build an algorithm $\mathcal{B}$ who can distinguish between $\mathcal{O}^{Sel}_0$ and $\mathcal{O}^{Sel}_1$ with ϵ using A.

Proof: This proof is identical with the proof of the previous lemma except the order between the k^{th} query and the challenge ciphertext and the oracles that $\mathcal{B}$ works with. Since this lemma simulates the selective security, the k^{th} key is requested after the challenge ciphertext is queried. ($k > q_1$) The algorithm works with either $\mathcal{O}^{Sel}_0$ or $\mathcal{O}^{Sel}_1$. If the simulator works with $\mathcal{O}^{Sel}_0$, $\mathcal{B}$ simulates $\textbf{SFKeyGen}(x_k, MSK, \boldsymbol{h}', m_2, 0)$. Therefore, this properly simulates G^N_{k,m_2}. If the simulator works with $\mathcal{O}_1^{Sel}$, $\mathcal{B}$ simulates $\mathbf{SFKeyGen}(x_k, MSK, \boldsymbol{h}', m_2, \alpha')$. This properly simulates G^T_{k,m_2}. □

Lemma 5. 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 $\mathcal{A}$.
Proof: The proof of this lemma is identical with that of Lemma 2 except that the kth key is temporary semi-functional (TSF) and outputs from either $\mathbf{SFKeyGen}(x, MSK, \boldsymbol{h}', j-1, \alpha_k)$ or $\mathbf{SFKeyGen}(x, MSK, \boldsymbol{h}', j, \alpha_k)$ where α'_k is a random value from $\mathbb{Z}_p$. The simulator can randomly select α'_i and use it to simulate the games. Hence, If $T = f_2^{cw}$, this simulate $\mathsf{G}^T_{k,j-1}$. Otherwise, if T is a random, this simulate $\mathsf{G}^T_{k,j}$. □

Lemma 6. 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 $\mathcal{A}$.
Proof: The proof of this lemma is providing in the supplementary material. □

6 Instance

We introduce a Non-monotonic Ciphertext-Policy Attribute-Based Encryption (NM-CP-ABE) with short keys and a Non-monotonic Key-Policy Attribute-Based Encryption (NM-KP-ABE) with short ciphertexts, an unbounded NM-CP-ABE and an unbounded NM-KP-ABE as new instances of our encodings. Particularly, we first introduce two NM-CP-ABE schemes, NM-CP-ABE with short keys unbounded NM-KP-ABE. Then, we convert them into corresponding NM-KP-ABE schemes using duality of ABE introduced in [8]. All our schemes are adaptively secure in prime order groups.

6.1 Adaptively Secure Unbounded NM-CP-ABE

Assumptions for NM-CP-ABE with multi-use of attributes We define assumptions A1-(n) and A2-(n) for our instances. Those assumptions are based on n-(A) and n-(B) assumptions introduced in [31], respectively. We prove the security of our assumptions in the generic group model in the supplementary material.

Assumption 4. *(A1-(n))* If a group generator $\mathcal{G}$ and a positive integer n are given, we define the following distribution

$$\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{R} \mathcal{G},\ c, d, x, y, z, a_1, ..., a_n, b_1, ..., b_n \xleftarrow{R} \mathbb{Z}_p,$$

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

where

$$\begin{array}{lrl} Z_1 = \{ & & dc, x, y, z, (dcz)^2, \\ & \forall i \in [n_1], & dcza_i, dcz/a_i, (dc)^2 za_i, y/a_i^2, y^2/a_i^2 \\ & \forall (i,j) \in [n_1, n_1], i \neq j, & dcza_i/a_j, dcyza_i/a_j^2, (dcz)^2 a_i/a_j, dcyb_i/b_j^2, dcyb_i^2/b_j^2 \\ & \forall (i,j) \in [n_1, n_1], & b_i, xb_i, b_ib_j, dcy/b_i^2, dcxy/b_i^2, dcxyb_i/b_j^2 \\ & \forall (i,j,k) \in [n_1, n_1, n_1], i \neq j\}, & dcyb_ib_j/b_k^2 \qquad \}, \\ Z_2 = \{ & \forall i \in [n_1], & a_i, dcz/a_i \\ & \forall (i,j) \in [n_1, n_1], i \neq j, & ya_i/a_j^2, zb_ib_j \\ & \forall (i,j) \in [n_1, n_1], & b_i, xb_i, xzb_i, zb_i, b_ib_j \qquad \}. \end{array}$$

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

Assumption 5. *(A2-(n))* If a group generator $\mathcal{G}$ and a positive integer n are given, we define following distribution

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

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

where

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

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

We define the advantage of an algorithm $\mathcal{A}$ in breaking *A1-(n)* and *A2-(n)* to be

$$Adv_{\mathcal{G},\mathcal{A}}^{\{A1\text{-}(n),A2\text{-}(n)\}}(\lambda) = |Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]|$$

Now, We introduce Unbounded NM-CP-ABE which utilizes two selective schemes Two mode Identity Based Broadcast Encryption (TIBBE) and NM-CP-ABE from Yamada et al. [31].

Encoding scheme for Unbounded-NM-CP-ABE Our encoding scheme for Unbounded-NM-CP-ABE consists of the following four algorithms:

Param(κ): It sets $\omega_1 = 1, \omega_2 = 7$ and $\omega = 11$. It selects $\alpha \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{w} = (\eta) \xleftarrow{R} \mathbb{Z}_p$, $\boldsymbol{h} = (\delta, \nu, \zeta, y_h, y_w, y_x, y_y) \xleftarrow{R} \mathbb{Z}_p^7$. It sets $\boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}) = (\eta,\ \eta \cdot y_x,\ \eta \cdot y_y,\ \delta,\ \nu,\ \zeta,\ y_h,\ y_w,\ y_x,\ y_y)$.

Enc$_1(S)$: The algorithm selects $r, r_0, r_1, ..., r_k \xleftarrow{R} \mathbb{Z}_p$. It then selects $r_1', ..., r_k' \xleftarrow{R} \mathbb{Z}_p$ such that $r = r_1' + ... + r_k'$ and sets $\boldsymbol{r} = (r_0, r_1, ..., r_k, r_1', ..., r_k')^{\dagger}$. It sets $d_1 = \alpha + \delta r + \nu r_0$, $d_2 = -r_0$, $d_3 = r$. For all $w_i \in S = \{w_1, ..., w_{|S|}\}$ such that S is not an empty set. It sets

$$d_{i,1} = -\zeta r + r_i(w_iy_h + y_w),\ d_{i,2} = r_i,\ d_{i,1}' = \eta r_i'(w_iy_x + y_y),\ d_{i,2}' = \eta r_i'$$

It defines $\boldsymbol{k}(\alpha, S, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r}) := (d_1, d_2, d_3, d_{i,1}, d_{i,2}, d_{i,1}', d_{i,2}'\ \forall i \in [|S|])$.

Enc$_2(\tilde{\mathbb{A}})$: For the non-monotonic access structure $\tilde{\mathbb{A}}$, there exists a monotonic 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 $\boldsymbol{c}(\tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s}) := (c_1, c_2, c_{i,1}, c_{i,2}, c_{i,3}\ \forall i \in [\ell])$ as follows:

$$c_{i,1} = \delta\lambda_i + \zeta t_i, \quad c_{i,2} = -t_i(x_iy_h + y_w), \quad c_{i,3} = t_i \quad \text{if } \rho(i) = x_i;$$

$$c_{i,1} = \delta\lambda_i + \eta y_x t_i, \quad c_{i,2} = -t_i(x_iy_x + y_y), \quad c_{i,3} = t_i \quad \text{if } \rho(i) = x_i'$$

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 an negated attribute).

† r is intentionally omitted since $r = r_1' + ... + r_k'$.

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 sets

$$\Lambda_i = c_{i,1} \cdot d_3 + c_{i,2} \cdot d_{\gamma,2} + c_{i,3} \cdot d_{\gamma,1} = \lambda_i \delta r \qquad \text{if } \rho(i) = x_i;$$

$$\Lambda_i = c_{i,1} \cdot d_3 + \sum_{j \in [1,k]} \left(\frac{c_{i,3} \cdot d'_{j,1} + c_{i,2} \cdot d'_{j,2}}{x_i - w_j} \right) = \lambda_i \delta r \quad \text{if } \rho(i) = x'_i.$$

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

Remark 2. In our scheme, $\boldsymbol{w}$ which causes non-linearity is corresponding to b in the selectvely secure NM-CP-ABE of [31]. One may think η is redundant in the construction, but it is essential in the security proof both in ours and [31].

We can prove the *computation α hiding* using Yamada et al.'s selective security proofs of Two-mode Identity Based Broadcast Encryption (TIBBE) and NM-CP-ABE [31]. The major difference in our proofs is that we set η which causes non-linearity independently from any information given by the adversary. Therefore, we can include g_1^η and g_2^η along with g_1 and g_2 in the initial instance. This causes some change in the selective proofs, but we still can use the many parts of the selective proofs because we still can select the rest of public parameters after seeing the target predicate. We prove *computational α hiding* formally in Lemmas 7 and 8. It should be noted that the other properties trivially holds in our encodings of NM-CP-ABE.

Lemma 7 *Suppose there exists a PPT adversary $\mathcal{A}$ who can distinguish between $\mathcal{O}_0^{Cos}(n_1)$ and $\mathcal{O}_1^{Cos}(n_1)$ with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ breaking A1-(n_1) with ϵ using $\mathcal{A}$ with an attributes set of size $k \leq n_1$.*
Proof: The proof of this lemma is providing in the supplementary material. □

Lemma 8. *Suppose there exists an $\mathcal{A}$ who can distinguish between two oracles $\mathcal{O}_0^{Sel}(n_2)$ and $\mathcal{O}_1^{Sel}(n_2)$ with a non-negligible advantage ϵ. Then, we can build $\mathcal{B}$ breaking A2-(n_2) with ϵ using $\mathcal{A}$ with an access matrix of size $\ell \times m$ where $\ell, m \leq n_2$.*
Proof: The proof of this lemma is providing in the supplementary material. □

6.2 Adaptively Secure NM-CP-ABE with short keys

We introduced an NM-CP-ABE with short keys. The co-selective security proved in Lemma 9 is inspired by the selective NM-KP-ABE scheme of [5].

Assumptions for NM-CP-ABE with short keys We define n-DBDHE in asymmetric pairing a for our instances. We prove the security of our assumptions in the generic group model in the supplementary material.

Assumption 6. *((Asymmetric) n-DBDHE)* If a group generator $\mathcal{G}$ and a positive integer n are given, we define the following distribution

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

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

$$\text{where } Z_1 = Z_2 := \{dc, b^i |\ \forall i \in [2n], i \neq n+1\}.$$

Given the instances, 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 $\mathcal{A}$ in breaking *n-DBDHE* to be

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

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 $P(z) = \prod_{y \in S}(z-y)$ 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-monotonic access structure $\tilde{\mathbb{A}}$, there exists a monotonic 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 an 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 $c_1 \cdot d_1 + c_2 \cdot d_2 - \prod_{i \in [I]} \mu_i \Lambda_i = \alpha s$.

The computational α hiding of our scheme can be proved by the following two lemmas.

Lemma 9. *Suppose there exists a PPT adversary $\mathcal{A}$ who can distinguish between $\hat{\mathcal{O}}_0^{Cos}(n_1)$ and $\hat{\mathcal{O}}_1^{Cos}(n_1)$ with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ breaking (Asymmetric) $n_1 - DBDHE$ with ϵ using $\mathcal{A}$ with an attributes set of size $k < n_1$ $(N = n_1)$.*

Proof: The proof of this lemma is providing in the supplementary material. □

Lemma 10. *Suppose there exists an $\mathcal{A}$ who can distinguish between two oracles $\hat{\mathcal{O}}_0^{Sel}(n_2)$ and $\hat{\mathcal{O}}_1^{Sel}(n_2)$ with a non-negligible advantage ϵ. Then, we can build $\mathcal{B}$ breaking A2-(n_2) with ϵ using $\mathcal{A}$ with an access matrix of size $\ell \times m$ where $\ell, m \leq n_2$.*

Proof: The proof of this lemma is providing in the supplementary material. □

6.3 Duality

We introduce Unbounded NM-KP-ABE and NM-KP-ABE with short keys as a dual scheme of our NM-CP-ABE schemes. Attrapadung and Yamada [8] showed that pair encoding schemes can be easily converted to their dual scheme (e.g. CP-ABE to KP-ABE) when g_1^s can be parsed from $g^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}$. As our scheme satisfies this structural condition, our scheme also can be converted to NM-KP-ABE schemes using their technique.

$\mathsf{Param}(\kappa)$: 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 $\boldsymbol{h}' = (\pi, \boldsymbol{h})$.

$\mathsf{Enc}_1(\tilde{\mathbb{A}})$: It runs $\mathsf{Enc}_2(\tilde{\mathbb{A}})$ 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 $\boldsymbol{r}' = (s, \boldsymbol{s})$.

$\mathsf{Enc}_2(S)$: It creates $s' \xleftarrow{R} \mathbb{Z}_p$ and runs $\mathsf{Enc}_1(S)$ 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 $\boldsymbol{s}' = \boldsymbol{r}$.

$\mathsf{Pair}(S, \tilde{\mathbb{A}})$: $\mathsf{Pair}(S, \tilde{\mathbb{A}})$ 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 $\alpha s' = d_1' \cdot c_1' - kEc^\top$.

We can prove the *computational α hiding* of our NM-KP-ABE which is invariance between oracles $\tilde{\mathcal{O}}_\beta^{Cos}(n_2)$ and $\tilde{\mathcal{O}}_\beta^{Sel}(n_1)$ using the oracles of NM-CP-ABE, $\mathcal{O}_\beta^{Cos}(n_1)$ and $\mathcal{O}_\beta^{Sel}(n_2)$.

Lemma 11. *Suppose there exists an $\mathcal{A}$ who can distinguish between two oracles $\tilde{\mathcal{O}}_0^{Cos}(n_1)$ and $\tilde{\mathcal{O}}_1^{Cos}(n_1)$ with a non-negligible advantage ϵ. Then, we can build $\mathcal{B}$ distinguishing between $\mathcal{O}_0^{Sel}(n_1)$ and $\mathcal{O}_1^{Sel}(n_1)$ with ϵ using $\mathcal{A}$ with an access matrix of size $\ell \times m$ where $\ell, m \leq n_1$.*

Proof: To simulate $\tilde{\mathcal{O}}_\beta^{Col}(n_1)$, the simulator selects $s', \pi' \xleftarrow{R} \mathbb{Z}_p$. To generate the initial instance, it requests an initial instance to $\mathcal{O}_\beta^{Sel}(n_1)$. It sets $\boldsymbol{b}'(\boldsymbol{w}',1,\boldsymbol{1}) = (1, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{1}))$. For the k-type response, the simulate requests the c-type response to $\mathcal{O}_\beta^{Cos}(n_1)$ to receive $\boldsymbol{c}$ and constructs the response $\boldsymbol{k}' := (d_1'(= \pi' s), \boldsymbol{c})$. To respond c-type query, it requests the k-type response to $\mathcal{O}_\beta^{Sel}(n_1)$ to receive $\boldsymbol{k}$. It outputs $\boldsymbol{c}' := (s', \tilde{\boldsymbol{k}})$ where $\tilde{\boldsymbol{k}} = \boldsymbol{k}(\alpha', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); \boldsymbol{r}) + \boldsymbol{k}(\pi' s', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0}); \boldsymbol{0})$[‡]

If $\beta = 0$, it sets $\pi = \pi'$ and the simulator properly simulate $\tilde{\mathcal{O}}_0^{Col}(n_1)$. Otherwise, if $\beta = 1$, it implicitly sets $\pi = \alpha'/s' + \pi'$. This implies $d_1' = \pi s - \alpha' s/s'$ and this properly simulate $\tilde{\mathcal{O}}_1^{Col}(n_1)$ where $\alpha = -\alpha' s/s'$ since α' is randomly distributed to the adversary. □

References

1. S. Agrawal and M. Chase. A study of pair encodings: Predicate encryption in prime order groups. In E. Kushilevitz and T. Malkin, editors, *TCC 2016-A*, volume 9563 of *LNCS*, pages 259–288, 2016.
2. N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In P. Q. Nguyen and E. Oswald, editors, *EUROCRYPT*, volume 8441 of *LNCS*, pages 557–577. Springer, 2014.
3. N. Attrapadung. Dual system encryption framework in prime-order groups. *IACR Cryptology ePrint Archive*, 2015:390, 2015.
4. N. Attrapadung. Dual system encryption framework in prime-order groups via computational pair encodings. In J. H. Cheon and T. Takagi, editors, *ASIACRYPT*, volume 10032 of *LNCS*, pages 591–623, 2016.
5. N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Ràfols. Attribute-based encryption schemes with constant-size ciphertexts. *Theor. Comput. Sci.*, 422:15–38, 2012.
6. N. Attrapadung and H. Imai. Conjunctive broadcast and attribute-based encryption. In H. Shacham and B. Waters, editors, *Pairing*, volume 5671 of *LNCS*, pages 248–265. Springer, 2009.

‡ $\boldsymbol{k}(\pi' s', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0}); \boldsymbol{0})$ equals $\boldsymbol{k}(\pi' s', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); \boldsymbol{0})$ by the *parameter vanishing* property. Hence, by *linearity in random variables.*, $\tilde{\boldsymbol{k}} = \boldsymbol{k}(\alpha' + \pi' s', \tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); \boldsymbol{r})$.

7. N. Attrapadung and B. Libert. Functional encryption for inner product: Achieving constant-size ciphertexts with adaptive security or support for negation. In *PKC*, volume 6056 of *LNCS*, pages 384–402. Springer, 2010.
8. 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.
9. A. Beimel. *Secure schemes for secret sharing and key distribution.* PhD thesis, PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.
10. D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, pages 258–275, 2005.
11. 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.
12. 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.
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. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
15. J. Kim, W. Susilo, F. Guo, and M. H. Au. A tag based encoding: An efficient encoding for predicate encryption in prime order groups. In V. Zikas and R. D. Prisco, editors, *SCN*, volume 9841 of *LNCS*, pages 3–22, 2016.
16. J. Kim, W. Susilo, F. Guo, and M. H. Au. Functional encryption for computational hiding in prime order groups via pair encodings. *Designs, Codes and Cryptography*, pages 1–24, 2017.
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, A. Sahai, and B. Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*, pages 273–285, 2010.
20. A. B. Lewko and B. Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. *IACR Cryptology ePrint Archive*, 2009:482, 2009.
21. 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.
22. 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.
23. Z. Liu and D. S. Wong. Practical ciphertext-policy attribute-based encryption: Traitor tracing, revocation, and large universe. In T. Malkin, V. Kolesnikov, A. B. Lewko, and M. Polychronakis, editors, *ACNS*, volume 9092 of *LNCS*, pages 127–146. Springer, 2015.
24. T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In M. Matsui, editor, *ASIACRYPT*, volume 5912 of *LNCS*, pages 214–231. Springer, 2009.
25. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In T. Rabin, editor, *CRYPTO*, volume 6223 of *LNCS*, pages 191–208. Springer, 2010.
26. T. Okamoto and K. Takashima. 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.
27. R. Ostrovsky, A. Sahai, and B. Waters. Attribute-based encryption with non-monotonic access structures. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 195–203. ACM, 2007.
28. S. C. Ramanna, S. Chatterjee, and P. Sarkar. Variants of waters' dual system primitives using asymmetric pairings - (extended abstract). In M. Fischlin, J. A. Buchmann, and M. Manulis, editors, *PKC*, volume 7293 of *Lecture Notes in Computer Science*, pages 298–315. Springer, 2012.

29. 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.
30. H. Wee. Dual system encryption via predicate encodings. In Y. Lindell, editor, *TCC*, volume 8349 of *LNCS*, pages 616–637. Springer, 2014.
31. S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. In *PKC*, volume 8383 of *LNCS*, pages 275–292. Springer, 2014.
32. S. Yamada, N. Attrapadung, G. Hanaoka, and N. Kunihiro. A framework and compact constructions for non-monotonic attribute-based encryption. *IACR Cryptology ePrint Archive*, 2014:181, 2014.

Appendix

A Proofs of Lemmas

Lemma 6. 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 $\mathcal{A}$.

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: $\mathcal{B}$ runs Param to get $\omega_1, \omega_2, \omega$ and a sequence of variables and functions to set $\boldsymbol{b}$. It randomly selects $y_g, y_u, y_v \in \mathbb{Z}_p, \boldsymbol{w} \in \mathbb{Z}_p^{\omega_1}, \boldsymbol{h} \in \mathbb{Z}_p^{\omega_2}$ sets $\alpha = ac$, $a = a$, $b = 1/y_g$ and $\tau = y_v + ay_u$. $\mathcal{B}$ publishes the public parameters

$$g_1 = f_1^{y_g}, g_1^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = {f_1}^{y_g \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, g_1^a = {f_1^a}^{y_g}, g_1^{a\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = {f_1^a}^{y_g \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})},$$

$$g_1^\tau = f_1^{y_g y_v} + (f_1^a)^{y_g y_u}, g_1^{\tau \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = f_1^{y_g y_v \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}(f_1^a)^{y_g y_u \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})},$$

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

It also sets $g_2^\alpha = f_2^{acy_g}, g_2 = f_2^{y_g}, g_2^{\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})} = {f_2}^{y_g \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h})}, v_2 = f_2^{y_v}, u_2 = f_2^{y_u}, f_2$. It should be noted that $\mathcal{B}$ sets g_2^α only 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: $\mathcal{B}$ runs Enc1 to get a sequence of coefficient of $\boldsymbol{k}$. For each key SK_i, $\mathcal{B}$ randomly selects $\alpha_i'' \in \mathbb{Z}_p$, $\boldsymbol{z} \in \mathbb{Z}_p^{m_k}$ and $\boldsymbol{r} \in \mathcal{R}_s$ and sets $\alpha_i' = y_g c + \alpha_i''$.

$$\boldsymbol{K}_0 = f_2^{y_g \boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})} v_2^{\boldsymbol{z}} (f_2^a)^{\boldsymbol{k}(-\alpha_i'',x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0});\boldsymbol{r})},$$

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

This is a properly distributed key since

$$\begin{aligned}
\boldsymbol{K}_0 &= (f_2^a)^{\boldsymbol{k}(-\alpha_i'',x,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{0});\boldsymbol{r})} f_2^{y_g \boldsymbol{k}(0,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})} v_2^{\boldsymbol{z}}, \\
&= f_2^{\boldsymbol{k}(y_g ac,x,\boldsymbol{b}(\boldsymbol{w},y_g,y_g\boldsymbol{h});\boldsymbol{r})} \cdot v_2^{\boldsymbol{z}} \cdot f_2^{\boldsymbol{k}(-y_g ac - a\alpha_i'',x,\boldsymbol{b}(\boldsymbol{w},y_g,y_g\boldsymbol{h});\boldsymbol{0})} &\quad (12)\\
&= g_2^{\boldsymbol{k}(\alpha,x,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})} \cdot v_2^{\boldsymbol{z}} \cdot f_2^{-a\boldsymbol{k}(\alpha_i',x,\boldsymbol{b}(\boldsymbol{w},0,0);\boldsymbol{0})} &\quad (13)
\end{aligned}$$

The equality of (12) holds due to linearity over random variables. Also, the equality of (13) holds by *parameter vanishing* and *relaxed linearity over hidden common variables.*

Challenge: When the adversary requests the challenge ciphertexts with two messages M_0 and M_1. $\mathcal{B}$ runs Enc1 to get a sequence of coefficient of $\boldsymbol{c}$. It randomly selects $\beta \in \{0,1\}$, $s'' \in \mathbb{Z}_p$, $\boldsymbol{s}, \boldsymbol{s}'' \in \mathcal{R}_s$ and $\boldsymbol{h}'' \in \mathbb{Z}_p^n$ and sets $s = d$, $s' = -y_g ad + as''$ and $\boldsymbol{s}' = a \cdot \boldsymbol{s}''$. 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:

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

$$\boldsymbol{C}_1 = (f_1^a)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s'',y_g\boldsymbol{s}+\boldsymbol{s}'')} f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');s'',\boldsymbol{s}'')} (f_1^d)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');-y_g,\boldsymbol{0})},$$

$$\boldsymbol{C}_2 = \boldsymbol{C}_0^{y_v} \boldsymbol{C}_1^{y_u}$$

This implicitly sets $\boldsymbol{h}' = \boldsymbol{h} + a^{-1}\boldsymbol{h}''$ (i.e. $\boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}') = \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}) + \boldsymbol{b}(\boldsymbol{w}, 1, a^{-1}\boldsymbol{h}'')$. Also, the ciphertext is properly distributed since

$$\begin{aligned}
\boldsymbol{C}_1 &= (f_1^a)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s'',y_g\boldsymbol{s}+\boldsymbol{s}'')} f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');s'',\boldsymbol{s}'')} (f_1^d)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');-y_g,\boldsymbol{0})} \\
&= (f_1^a)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});y_g d - y_g d + s'',y_g\boldsymbol{s}+\boldsymbol{s}'')} f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}'');-y_g d+s'',\boldsymbol{s}'')} \qquad (14) \\
&= (f_1^a)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});y_g d,y_g\boldsymbol{s})} f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});-y_g ad+as'',a\boldsymbol{s}'')} \\
&\quad \cdot f_1^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},0,\boldsymbol{h}''/a);-y_g ad+as'',a\boldsymbol{s}'')} \qquad (15) \\
&= (g_1^a)^{\boldsymbol{c}(y,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});d,\boldsymbol{s})} (f_1)^{\boldsymbol{c}(y,(\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}+\boldsymbol{h}''/a);-y_g ad+as'',a\boldsymbol{s}'')} \qquad (16)
\end{aligned}$$

The equalities of (14) and (15) hold by linearity over random variables. The equality of (16) holds because of *relaxed linearity over hidden common parameters*. It should be noted that $\boldsymbol{h}''$ does not appear anywhere else, it only used in the challenge ciphertext. Hence, $\boldsymbol{h}'$ 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}. □

Lemma 7 *Suppose there exists a PPT adversary $\mathcal{A}$ who can distinguish between $\mathcal{O}_0^{Cos}(n_1)$ and $\mathcal{O}_1^{Cos}(n_1)$ with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ breaking A1-(n_1) with ϵ using $\mathcal{A}$ with an attributes set of size $k \leq n_1$.*

Proof: Given $D := \{g_1, g_2, g_1^c, g_2^c\} \cup \{g_1^{z_1}, g_2^{z_2} | z_1 \in Z_1, z_2 \in Z_2\}$ and T_β where from A1-(n_1). $\mathcal{B}$ will

$$\begin{aligned}
Z_1 = \{ \quad & dc, x, y, z, (dcz)^2, \\
& \forall i \in [n_1],\ dcza_i, dcz/a_i, (dc)^2 za_i, y/a_i^2, y^2/a_i^2 \\
& \forall (i,j) \in [n_1, n_1], i \neq j,\ dcza_i/a_j, dcyza_i/a_j^2, (dcz)^2 a_i/a_j, dcyb_i/b_j^2, dcyb_i^2/b_j^2 \\
& \forall (i,j) \in [n_1, n_1],\ b_i, xb_i, b_ib_j, dcy/b_i^2, dcxy/b_i^2, dcxyb_i/b_j^2 \\
& \forall (i,j,k) \in [n_1, n_1, n_1], i \neq j\},\ dcyb_ib_j/b_k^2 \qquad \},
\end{aligned}$$

$$\begin{aligned}
Z_2 = \{ \quad & \forall i \in [n_1],\ a_i, dcz/a_i \\
& \forall (i,j) \in [n_1, n_1], i \neq j,\ ya_i/a_j^2, zb_ib_j \\
& \forall (i,j) \in [n_1, n_1],\ b_i, xb_i, xzb_i, zb_i, b_ib_j \qquad \},
\end{aligned}$$

simulate either $\mathcal{O}_0^{Cos}(n_1)$ or $\mathcal{O}_1^{Cos}(n_1)$ using $\mathcal{A}$.

Initial response When $\mathcal{A}$ sends the initial query to $\mathcal{B}$, $\mathcal{B}$ implicitly sets $\eta = \sum_{i\in[n_1]} b_i$ and computes $g_1^\eta = \prod_{i\in[n_1]} g_1^{b_i}$ and $g_2^\eta = \prod_{i\in[n_1]} g_2^{b_i}$. It outputs $\{g_1, g_2, g_1^\eta, g_2^\eta\}$.

k-type response When $\mathcal{A}$ sends the k-type query for a set of attributes $S^* = (w_1^*, ..., w_k^*)$ where $k \leq n_1$ to $\mathcal{B}$. $\mathcal{B}$ then randomly selects $\tilde{\zeta}$, ν', t', $\tilde{h}$, $\tilde{w}$, $\tilde{y}$ from $\mathbb{Z}_p$ and sets $\delta = dy$ and $\zeta = dc + \tilde{\zeta}$. Then, it also sets

$$y_h = \tilde{h} + \sum_{i\in[k]} y/a_i^2,\ y_w = \tilde{w} + \sum_{i\in[k]} (dcz/a_i - w_i^* y/a_i^2),\ y_x = x + \sum_{i\in[k]} b_i,$$

$$y_y = \tilde{y} - \sum_{i\in[k]} w_ib_i,\ \eta y_x = x\sum_{i\in[n_1]} b_i + \sum_{i,j\in[k,n_1]} b_ib_j,\ \eta y_y = \tilde{y}\sum_{i\in[n_1]} b_i - \sum_{i,j\in[k,n_1]} w_ib_ib_j.$$

y_x is properly distributed due to x which is used only for y_x.

$\mathcal{B}$ selects $r_0 \xleftarrow{R} \mathbb{Z}_p$ and implicitly sets $r = z$. Allocating z to r is hidden to $\mathcal{A}$ since z was used nowhere else. To compute $g_2^{\boldsymbol{k}(\beta\cdot\alpha,S^*,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})}$, $\mathcal{B}$ sets

$$g_2^{d_1} = T \cdot g_2^{\nu r_0}, \quad g_2^{d_2} = g_2^{r_0}, \quad g_2^{d_3} = g_2^z.$$

If $T = g_2^{dyz}$, then $g_2^{d_1} = g_2^{\delta r + \nu r_0}$. Therefore, $\mathcal{B}$ simulates the k-type response of $\mathcal{O}_0^{Cos}(n_1)$. If T is random from G_2 and we denote it $T = g_2^{dyz} g_2^\alpha$, $g_2^{d_1} = g_2^{\alpha+\delta r+\nu r_0}$, $\mathcal{B}$ simulates the k-type outputs of $\mathcal{O}_1^{Cos}(n_1)$.

For all $\psi \in [k]$, $\mathcal{B}$ computes the following:

• **Compute** $g_2^{d_{\psi,1}}$ and $g_2^{d_{\psi,2}}$: For $w_\psi^* \in S^*$, $\mathcal{B}$ randomly chooses $\hat{r}_\psi$ and sets $r_\psi = a_\psi - \hat{r}_\psi$. It computes $g_2^{d_{\psi,1}}$ and $g_2^{d_{\psi,2}}$ as follows:

$$\begin{aligned}
g_2^{d_{\psi,1}} &= g_2^{-(dc+\tilde{\zeta})z + r_\psi(w_\psi^* y_h + y_w)} \\
&= g_2^{\boxed{-dcz} - \tilde{\zeta}z + w_\psi^* \tilde{h} a_\psi + w_\psi^* a_\psi \sum_{i\in[k]} y/a_i^2} \\
&\quad \cdot g_2^{a_\psi \tilde{w} + a_\psi \sum_{i\in[k]}(\boxed{dcz}/a_i - w_i^* y/a_i^2) - \hat{r}_\psi(w_\psi^* y_h + y_w)} \\
&= (g_2^z)^{-\tilde{\zeta}} (g_2^{a_\psi})^{w_\psi^* \tilde{h} + \tilde{w}} \prod_{i\in[k], i\neq\psi} (g_2^{y a_\psi/a_i^2})^{(w_\psi^* - w_i^*)} \\
&\quad \cdot \prod_{i\in[k], i\neq\psi} g_2^{dcz/a_i} g_2^{-\hat{r}_\psi(w_\psi^* y_h + y_w)}, \\
g_2^{d_{\psi,2}} &= g_2^{a_\psi} g_2^{-\hat{r}_\psi}.
\end{aligned}$$

• **Compute** $g_2^{d'_{\psi,1}}$ and $g_2^{d'_{\psi,2}}$: For $w_\psi^* \in S^*$, $\mathcal{B}$ randomly chooses $\hat{r}'_1, ..., \hat{r}'_k$ such that $\sum_{i\in[k]} \hat{r}'_i = 0$ and implicitly sets $r'_i = \hat{r}'_i + zb_i/\eta$ for all $i \in [k-1]$ and $r'_k = \hat{r}'_k + \sum_{i\in[n_1]\setminus[k-1]} zb_i/\eta$. This sets $\sum_{i\in[k]} r'_i = \sum_{i\in[k]} \hat{r}'_i + \sum_{i\in[n_1]} zb_i/\eta = 0 + z\eta/\eta = z$. All r'_i are distributed randomly due to $\hat{r}'_i$.

[**Case 1**] For $\psi \in [k-1]$, $\mathcal{B}$ computes and

$$\begin{aligned}
g_2^{d'_{\psi,1}} &= g_2^{(\hat{r}'_\psi + zb_\psi/\eta)\eta(w_\psi^* x + w_\psi^* \sum_{i\in[k]} b_i + \tilde{y} - \sum_{i\in[k]} w_i^* b_i)} \\
&= g_2^{\hat{r}'_\psi(w_\psi^* \eta x + \eta\tilde{y} - \eta\sum_{i\in[k], i\neq\psi}(w_\psi^* - w_i^*)b_i) + (w_\psi^* xzb_\psi + \tilde{y}zb_\psi - \sum_{i\in[k], i\neq\psi}(w_\psi^* - w_i^*)zb_i b_\psi)} \\
&= \prod_{i\in[n_1]} g_2^{xb_i}{}^{\hat{r}'_\psi w_\psi^*} \prod_{i\in[n_1]} g_2^{b_i}{}^{\hat{r}'_\psi \tilde{y}} \prod_{\substack{(i,j)\in[k,n_1] \\ i\neq\psi}} g_2^{b_i b_j}{}^{-(w_\psi^* - w_i^*)} g_2^{\Phi_1}, \\
g_2^{d'_{\psi,2}} &= g_2^{\eta(\hat{r}'_\psi + b_\psi z/\eta)} = (\prod_{i\in[n_1]} g_2^{b_i}{}^{\hat{r}'_\psi}) \cdot g_2^{zb_\psi}
\end{aligned}$$

where $\Phi_1 = w_\psi^* xzb_\psi + \tilde{y}zb_\psi - \sum_{i\in[k], i\neq\psi}(w_\psi^* - w_i^*)zb_i b_\psi$. $g_2^{\Phi_1}$ can be computed since $g_2^{xzb_\psi}$, $g_2^{zb_\psi}$ and $g_2^{zb_i b_\psi}$ are given in the instance.

[**Case 2**] For $\psi = k$, $\mathcal{B}$ computes $g_2^{d'_{k,1}}$ and $g_2^{d'_{k,2}}$ as follows:

$$\begin{aligned}
g_2^{d'_{k,1}} &= g_2^{(\hat{r}'_k + \sum_{i\in[n_1]\setminus[k-1]} zb_i/\eta)(w_k^* \eta x + w_k^* \eta \sum_{i\in[k]} b_i + \tilde{y}\eta - \eta\sum_{i\in[k]} w_i^* b_i)} \\
&= g_2^{\Phi_2} \cdot g_2^{w_k^* \sum_{i\in[n_1]\setminus[k-1]} xzb_i + \tilde{y}\sum_{i\in[n_1]\setminus[k-1]} zb_i - \sum_{i\in[k-1], j\in[n_1]\setminus[k-1]}(w_\psi^* - w_i^*)zb_i b_j} \\
&= g_2^{\Phi_2} \prod_{i\in[n_1]\setminus[k-1]} g_2^{xzb_i}{}^{w_k^*} \prod_{i\in[n_1]\setminus[k-1]} g_2^{zb_i}{}^{\tilde{y}} \prod_{i\in[k-1], j\in[n_1]\setminus[k-1]} g_2^{zb_i b_j}{}^{-(w_\psi^* - w_i^*)}, \\
g_2^{d'_{k,2}} &= g_2^{\eta(\hat{r}'_\psi + \sum_{i\in[n_1]\setminus[k-1]} b_\psi z/\eta)} = g_2^{\eta\hat{r}'_\psi + \sum_{i\in[n_1]\setminus[k-1]} zb_\psi} = \prod_{i\in[n_1]} g_2^{b_i}{}^{\hat{r}'_\psi} \prod_{i\in[n_1]\setminus[k-1]} g_2^{zb_i}
\end{aligned}$$

where $\Phi_2 = \hat{r}'_k(w_k^* \eta x + \eta\tilde{y} - \eta\sum_{i\in[k-1]}(w_k^* - w_i^*)b_i)$. $g_2^{\Phi_2}$ can be computed since $g_2^{xb_i}$, $g_2^{b_i}$ and $g_2^{b_i b_j}$ are given. Finally, it outputs $g_2^{\boldsymbol{k}(\beta\cdot\alpha, S^*, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); \boldsymbol{r})}$.

c-type response When $\mathcal{A}$ sends c-type query for $\tilde{\mathbb{A}}$ to $\mathcal{B}$ after it sent the k-type query. Since S^* of k-type query does not satisfy $\tilde{\mathbb{A}} = NM(\mathbb{A})$, there exists $S' = N(S^*) \notin \mathbb{A}$ where $\mathbb{A} = (A, \rho)$ and A is an $\ell \times m$ matrix. We let A_ψ is the ψth row of A. By proposition 11 in [13], $\mathcal{B}$ can select a random vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_m) \in \mathbb{Z}_p^m$ which satisfies $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$ for all i' such that

$\rho(i') \in S'$ and $\langle(1,0,...0),\boldsymbol{\theta}\rangle = 1$. It then implicitly sets $\boldsymbol{\phi} = (s, s_2, ..., s_m) = c \cdot \boldsymbol{\theta} + (dy)^{-1}\boldsymbol{\mu}$ where $\boldsymbol{\mu} = (0, \mu_2, ..., \mu_m)$ is randomly selected from $\mathbb{Z}_p^m$. This implicitly sets $s = c$. c was already used in ζ, but its value was not revealed because of $\tilde{\zeta}$ which only appears in ζ. Therefore, assigning c to s is hidden to the adversary. The other values $s_2, ..., s_m$ are randomly distributed due to $\boldsymbol{\mu}$.

To compute the c-type response $g_1^{\boldsymbol{c}(\tilde{\mathbb{A}},\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});s,\boldsymbol{s})}$, it sets $g_1^{c_1} = (g_1^c)$ and $g_1^{c_2} = (g_1^c)^\nu$. To compute the other elements, it also computes $g_1^{c_{i',1}}$, $g_1^{c_{i',2}}$ and $g_1^{c_{i',3}}$ for $i' \in [\ell]$ using one of following four cases.

[**Case 1**] $\rho(i')$ is not a negated attribute and $\rho(i') \in S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$).

$\mathcal{B}$ randomly selects $t_{i'} \in \mathbb{Z}_p$ and computes

$$
\begin{aligned}
g_1^{c_{i',1}} &= g_1^{\langle A_{i'},\boldsymbol{\mu}\rangle + \zeta t_{i'}} = g_1^{\langle A_{i'},\boldsymbol{\mu}\rangle + \tilde{\zeta} t_{i'}} (g_1^{dc})^{t_{i'}}, \\
g_1^{c_{i',2}} &= g_1^{-t_{i'}(\rho(i')\tilde{h} + \rho(i')\sum_{i\in[k]} y/a_i^2 + \tilde{w} + \sum_{i\in[k]}(dcz/a_i - w_i^* y/a_i^2))}, \\
&= g_1^{-t_{i'}(w_t\tilde{h}+\tilde{w})} (g_1^{dcz/a_i})^{-t_{i'}} (g_1^{y/a_i^2})^{-t_{i'}(\rho(i')-w_i^*)}. \\
g_1^{c_{i',3}} &= g_1{}^{t_{i'}}.
\end{aligned}
$$

[**Case 2**] $\rho(i')$ is not a negated attribute and $\rho(i') \notin S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle \neq 0$).

$\mathcal{B}$ selects $\hat{t}_{i'} \xleftarrow{R} \mathbb{Z}_p$ and sets $t_{i'} = \hat{t}_{i'} - \langle A_{i'}, \boldsymbol{\theta}\rangle y + \sum_{i\in[k]} \frac{\langle A_{i'},\boldsymbol{\theta}\rangle dcza_i}{\rho(i') - w_i^*}$. Then,

$$
\begin{aligned}
g_1^{c_{i',1}} &= g_1^{\delta\langle A_{i'},\boldsymbol{\phi}\rangle + \zeta t_{i'}} = g_1^{\boxed{\langle A_{i'},\boldsymbol{\theta}\rangle dcy} + \langle A_{i'},\boldsymbol{\mu}\rangle + dc\hat{t}_{i'} - \boxed{\langle A_{i'},\boldsymbol{\theta}\rangle dcy} + \sum_{i\in[k]} \frac{\langle A_{i'},\boldsymbol{\theta}\rangle (dc)^2 za_i}{\rho(i')-w_i^*} + \tilde{\zeta} t_{i'}} \\
&= g_1^{\langle A_{i'},\boldsymbol{\mu}\rangle} (g_1^{dc})^{\hat{t}_{i'}} \prod_{i\in[k]} (g_1^{(dc)^2 za_i})^{\frac{\langle A_{i'},\boldsymbol{\theta}\rangle}{\rho(i')-w_i^*}} \cdot (g_1^{c_{i',3}})^{\tilde{\zeta}} \\
g_1^{c_{i',2}} &= g_1^{-(\hat{t}_{i'} - \langle A_{i'},\boldsymbol{\theta}\rangle y + \sum_{i\in[k]} \frac{\langle A_{i'},\boldsymbol{\theta}\rangle dcza_i}{\rho(i')-w_i^*})\cdot(\rho(i')\tilde{h} + \rho(i')\sum_{j\in[k]} y/a_j^2 + \tilde{w} + \sum_{j\in[k]}(dcz/a_j - w_j^* y/a_j^2))} \\
&= g_1^{-\hat{t}_{i'}(\rho(i')\tilde{h} + \rho(i')\sum_{j\in[k]} y/a_j^2 + \tilde{w} + \sum_{j\in[k]}(dcz/a_j - w_j^* y/a_j^2))} \\
&\quad \cdot g_1^{\langle A_{i'},\boldsymbol{\theta}\rangle y(\rho(i')\tilde{h} + \rho(i')\sum_{j\in[k]} y/a_j^2 + \tilde{w} + \sum_{j\in[k]}(dcz/a_j - w_j^* y/a_j^2))} \\
&\quad \cdot g_1^{-\sum_{i\in[k]} \frac{\langle A_{i'},\boldsymbol{\theta}\rangle dcza_i}{\rho(i')-w_i^*}(\rho(i')\tilde{h} + \rho(i')\sum_{j\in[k]} y/a_j^2 + \tilde{w} + \sum_{j\in[k]}(dcz/a_j - w_j^* y/a_j^2))} \\
&= g_1^{\Phi_3} \cdot g_1^{\langle A_{i'},\boldsymbol{\theta}\rangle \left((\rho(i')\tilde{h}+\tilde{w})y + \sum_{j\in[k]}(\rho(i')-w_j^*)y^2/a_j^2 + \boxed{\sum_{j\in[k]} dcyz/a_j}\right)} \\
&\quad \cdot g_1^{-\langle A_{i'},\boldsymbol{\theta}\rangle \left((\rho(i')\tilde{h}+\tilde{w})\sum_{i\in[k]} \frac{dcza_i}{\rho(i')-w_i^*} + \boxed{\sum_{i\in[k]} \frac{dcza_i}{\rho(i') - w_i^*} \sum_{j\in[k]} (\rho(i') - w_j^*)y/a_j^2}\right)} \\
&\quad \cdot g_1^{-\langle A_{i'},\boldsymbol{\theta}\rangle \left(\sum_{i\in[k]} \frac{dcza_i}{\rho(i')-w_i^*} \sum_{j\in[k]} dcz/a_j\right)} \\
&= g_1^{\Phi_3} \cdot g_1^{\Phi_4} \cdot \prod_{\substack{(i,j)\in[k,k] \\ i\neq j}} (g_1^{\frac{dcyza_i}{a_j^2}})^{\frac{-\langle A_{i'},\boldsymbol{\theta}\rangle(\rho(i')-w_j^*)}{\rho(i')-w_i^*}} \prod_{(i,j)\in[k,k]} (g_1^{\frac{(dcz)^2 a_i}{a_j}})^{\frac{\langle A_{i'},\boldsymbol{\theta}\rangle}{\rho(i')-w_i^*}} \\
g_1^{c_{i',3}} &= g_1^{t_{i'}} = g_1^{\hat{t}_{i'}} (g_1^y)^{-\langle A_{i'},\boldsymbol{\theta}\rangle} \prod_{i\in[k]} (g_1^{dcza_i})^{\frac{\langle A_{i'},\boldsymbol{\theta}\rangle}{\rho(i')-w_i^*}}
\end{aligned}
$$

where $\Phi_3 = -\hat{t}_{i'}(\rho(i')\tilde{h} + \tilde{w}) - \hat{t}_{i'}\sum_{j\in[k]} dcz/a_j - \hat{t}_{i'}\sum_{j\in[k]}(\rho(i') - w_j^*)y/a_j^2$ and

$$
\Phi_4 = \langle A_{i'}, \boldsymbol{\theta}\rangle\big((\rho(i')\tilde{h} + \tilde{w})y + \sum_{j\in[k]}(\rho(i') - w_j^*)y^2/a_j^2 - (\rho(i')\tilde{h} + \tilde{w})\sum_{i\in[k]} \frac{dcza_i}{\rho(i') - w_i^*}\big).
$$

Therefore, $g_1^{\Phi_3}$ and $g_1^{\Phi_4}$ can be computed since g_1, g_1^{dcz/a_j}, g_1^{y/a_j^2}, g_1^{y}, $g_1^{y^2/a_j^2}$ and $g_1^{dcza_i}$ are given in the instance.

[**Case 3**] $\rho(i')$ is a negated attribute and $\rho(i') \in S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$).

$\mathcal{B}$ randomly selects $t_{i'}$ from $\mathbb{Z}_p$. We let w'_ψ denote $\rho(i')$. Because $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$, $\mathcal{B}$ can compute

$$g_1^{c_{i',1}} = g_1{}^{\langle A_{i'}, \boldsymbol{\mu}\rangle} \prod_{i\in[n_1]} (g_1^{xb_i})^{t_{i'}} \prod_{(i,j)\in[k,n_1]} (g_1^{b_ib_j})^{t_{i'}},$$

$$g_1^{c_{i',2}} = (g_1^{x})^{t_{i'}(w'_\psi+\tilde{y})} \prod_{i\in[k]} (g_1^{b_i})^{(w'_\psi - w_i)t_{i'}}, \qquad g_1^{c_{i',3}} = g_1^{t_{i'}}.$$

[**Case 4**] $\rho(i')$ is a negated attribute and $\rho(i') \notin S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle \neq 0$).

$\mathcal{B}$ randomly selects $\hat{t}_{i'}$ from $\mathbb{Z}_p$ and implicitly sets $t_{i'} = \hat{t}_{i'} - \langle A_{i'}, \boldsymbol{\theta}\rangle dcy/b_\psi^2$. $t_{i'}$ is properly distributed due to the random value $\hat{t}_{i'}$. $\mathcal{B}$ computes

$$\begin{aligned}
g_1^{c_{i',1}} &= g_1^{\langle A_{i'},\boldsymbol{\theta}\rangle dcy + \langle A_{i'},\boldsymbol{\mu}\rangle + (\hat{t}_{i'} - \langle A_{i'},\boldsymbol{\theta}\rangle dcy/b_\psi^2)(\sum_{i\in[n_1]} xb_i + \sum_{(i,j)\in[k,n_1]} b_ib_j)} \\
&= g_1^{\boxed{\langle A_{i'},\boldsymbol{\theta}\rangle dcy} + \langle A_{i'},\boldsymbol{\mu}\rangle + \hat{t}_{i'}(\sum_{i\in[n_1]} xb_i + \sum_{(i,j)\in[k,n_1]} b_ib_j)} \\
&\quad \cdot g_1^{-\langle A_{i'},\boldsymbol{\theta}\rangle \sum_{i\in[n_1]} dcxyb_i/b_\psi^2 - \boxed{\langle A_{i'},\boldsymbol{\theta}\rangle \sum_{(i,j)\in[k,n_1]} dcyb_ib_j/b_\psi^2}} \\
&= g_1^{\langle A_{i'},\boldsymbol{\mu}\rangle} \prod_{i\in[n_1]} (g_1^{xb_i})^{\hat{t}_{i'}} \prod_{(i,j)\in[k,n_1]} (g_1^{b_ib_j})^{\hat{t}_{i'}} \prod_{i\in[n_1]} (g_1^{dcxyb_i/b_\psi^2})^{-\langle A_{i'},\boldsymbol{\theta}\rangle} \\
&\quad \prod_{\substack{(i,j)\in[k,n_1] \\ i,j\neq\psi}} (g_1^{dcyb_ib_j/b_\psi^2})^{-\langle A_{i'},\boldsymbol{\theta}\rangle}, \\
g_1^{c_{i',2}} &= g_1^{(\hat{t}_{i'} - \langle A_{i'},\boldsymbol{\theta}\rangle dcy/b_\psi^2)(w'_\psi x + w'_\psi \sum_{i\in[k]} b_i + \hat{y} - \sum_{i\in[k]} w_ib_i)} \\
&= g_1^{\hat{t}_{i'}(w'_\psi x + w'_\psi \sum_{i\in[k]} b_i + \tilde{y} - \sum_{i\in[k]} w_ib_i) - \langle A_{i'},\boldsymbol{\theta}\rangle (w'_\psi x + w'_\psi \sum_{i\in[k]} b_i + \tilde{y} - \sum_{i\in[k]} w_ib_i) dcy/b_\psi^2} \\
&= g_1^{\Phi_5} (g_1^{dcy/b_\psi^2})^{-\tilde{y}\langle A_{i'},\boldsymbol{\theta}\rangle} (g_1^{dcxy/b_\psi^2})^{\langle A_{i'},\boldsymbol{\theta}\rangle w'_\psi} \prod_{\substack{i\in[k] \\ i\neq\psi}} (g_1^{dcyb_i/b_\psi^2})^{-\langle A_{i'},\boldsymbol{\theta}\rangle (w'_\psi - w'_i)}, \\
g_1^{c_{i',3}} &= g_1^{\hat{t}_{i'}} (g_1^{dcy/b_\psi^2})^{-\langle A_{i'},\boldsymbol{\theta}\rangle}
\end{aligned}$$

where $\Phi_5 = \hat{t}_{i'}(w'_\psi x + w'_\psi \sum_{i\in[k]} b_i + \tilde{y} - \sum_{i\in[k]} w_ib_i)$. $g_1^{\Phi_5}$ can be computed using g_1, g_1^x and $g_1^{b_i}$. Therefore, if $T = g_2^{dyz}$, $\mathcal{B}$ simulates $\mathcal{O}_0^{Cos}(n_1)$. If T is a random from G_2, $\mathcal{B}$ simulates $\mathcal{O}_1^{Cos}(n_1)$. □

Lemma 8. *Suppose there exists an $\mathcal{A}$ who can distinguish between two oracles $\mathcal{O}_0^{Sel}(n_2)$ and $\mathcal{O}_1^{Sel}(n_2)$ with a non-negligible advantage ϵ. Then, we can build $\mathcal{B}$ breaking A2-(n_2) with ϵ using $\mathcal{A}$ with an access matrix of size $\ell \times m$ where $\ell, m \leq n_2$.*

Proof: Given $D := \{g_1, g_2, g_1^c, g_2^c\} \cup \{g_1^{z_1}, g_2^{z_2} | z_1 \in Z_1, z_2 \in Z_2\}$ and T_β where from A2-(n_2), $\mathcal{B}$ will simulate either $\mathcal{O}_0^{Sel}(n_2)$ or $\mathcal{O}_1^{Sel}(n_2)$ using $\mathcal{A}$.

Initial response: When the adversary $\mathcal{A}$ requests an initial instance to $\mathcal{B}$. $\mathcal{B}$ computes $g_1^\eta = g_2^{\sum_{i\in[n_2]} b_i}$ and $g_2^\eta = g_2^{\sum_{i\in[n_2]} b_i}$ and outputs $\{g_1, g_2, g_1^\eta, g_2^\eta\}$.

c-type response: When $\mathcal{A}$ sends the c-type query for an access policy $\tilde{\mathbb{A}}^* = NM(\mathbb{A}^*)$ where $\mathbb{A}^* = (A^*, \rho^*)$ and the access matrix A^* is an $\ell \times m$ matrix where $\ell, m \leq n_2$. We define two sets $L_N = \{i | i \in [\ell] \wedge \rho^*(i) = x'_i\}$ for negated attributes and $L_P = \{i | i \in [\ell] \cap \rho^*(i) = x_i\}$ for

$$
\begin{aligned}
Z_1 = \{ \quad & \forall(i,j) \in [n_2, n_2],\ dc, a, b_j, dcb_j, dcb_ib_j, a^i/b_j^2 \\
& \forall(i,j,j') \in [2n_2, n_2, n_2], j \neq j',\ a^ib_j/b_{j'}^2 \\
& \forall(i,j,j') \in [n_2, n_2, n_2], j \neq j',\ dca^ib_j/b_{j'}, dca^ib_j/b_{j'}^2 \\
& \forall(i,j,j',j'') \in [n_2, n_2, n_2, n_2], j \neq j', j' \neq j''\},\ dca^ib_jb_{j'}/b_{j''}^2 \quad \},
\end{aligned}
$$

$$
\begin{aligned}
Z_2 = \{ \quad & \forall(i,j) \in [n_2, n_2],\ dc, a^i, a^ib_j, a^i/b_j^2 \\
& \forall(i,j) \in [2n_2, n_2], i \neq n_2+1,\ a^i/b_j \\
& \forall(i,j,j') \in [2n_2, n_2, n_2], j \neq j',\ a^ib_j/b_{j'}^2 \quad \}
\end{aligned}
$$

non-negated attributes. $\mathcal{B}$ randomly selects $\tilde{h}$, $\tilde{w}$, $\tilde{x}$, $\tilde{y}, \tilde{\zeta}$, $\tilde{\nu}$ from $\mathbb{Z}_p$ and implictly sets

$$
\delta = da, \quad \zeta = \tilde{\zeta} + \sum_{(j,k) \in L_P \times [m]} A^*_{j,k} \cdot a^k/b_j, \quad \nu = d + \tilde{\nu},
$$

$$
y_h = \tilde{h} + \sum_{(j,k) \in L_P \times [m]} A^*_{j,k} a^k/b_j^2, \quad y_w = \tilde{w} - \sum_{(j,k) \in L_P \times [m]} \rho^*(j) A^*_{j,k} a^k/b_j^2,
$$

$$
y_x = \tilde{x} + \sum_{(j,k) \in L_N \times [m]} A^*_{j,k} a^k/b_j^2, \quad y_y = \tilde{y} - \sum_{(j,k) \in L_N \times [m]} \rho^*(j) A^*_{j,k} a^k/b_j^2,
$$

Since $g_1, g_1^{b_j}, g_1^{a^k/b_j}, g_1^{a^k/b_j^2}$ and $g_1^{a^kb_i/b_j^2}$ are given in the instance, $\mathcal{B}$ can compute

$$
g_1^{y_h} = g_1^{\tilde{h}} \prod_{(j,k) \in L_P \times [m]} (g_1^{a^k/b_j^2})^{A^*_{j,k}}, \quad g_1^{y_w} = g_1^{\tilde{w}} \prod_{(j,k) \in L_P \times [m]} (g_1^{a^k/b_j^2})^{-\rho^*(j) A^*_{j,k}},
$$

$$
g_1^{y_x} = g_1^{\tilde{x}} \prod_{(j,k) \in L_N \times [m]} (g_1^{a^k/b_j^2})^{A^*_{j,k}}, \quad g_1^{y_y} = g_1^{\tilde{y}} \prod_{(j,k) \in L_N \times [m]} (g_1^{a^k/b_j^2})^{-\rho^*(j) A^*_{j,k}},
$$

$$
\begin{aligned}
g_1^{\eta y_x} &= (\prod_{i \in [n_2]} g_1^{b_i})^{\tilde{x}} \prod_{(i,j,k) \in [n_2] \times L_N \times [m]} (g_1^{a^kb_i/b_j^2})^{A^*_{j,k}} \\
&= (\prod_{i \in [n_2]} g_1^{b_i})^{\tilde{x}} \prod_{\substack{(i,j,k) \in [n_2] \times L_N \times [m] \\ i \neq j}} (g_1^{a^kb_i/b_j^2})^{A^*_{j,k}} \prod_{(j,k) \in L_N \times [m]} (g_1^{a^k/b_j})^{A^*_{j,k}}.
\end{aligned}
$$

For a negated attribute w_i', we define $g_1^{w_i'} := g_1^{w_i}$ and $g_2^{w_i'} := g_2^{w_i}$ where $g_1 \in G_1$ and $g_2 \in G_2$. $\mathcal{B}$ randomly selects $\hat{s}_2, ..., \hat{s}_m$ from $\mathbb{Z}_p$ and implictly sets $\boldsymbol{\phi} = c(1, a, a^2, ..., a^{m-1}) + d^{-1}(0, \hat{s}_2, \hat{s}_3, ..., \hat{s}_m)$. This sets $s = c$. Due to $\hat{s}_2, \hat{s}_3, ..., \hat{s}_m$, s does not correlate to the other values in $\boldsymbol{\phi}$. For A^*_ψ where $\psi \in [\ell]$, $\mathcal{B}$ sets

$$
\lambda_\psi = \langle A^*_\psi, \boldsymbol{\phi} \rangle = \sum_{i \in [m]} A^*_{\psi,i} ca^{i-1} + d^{-1} \sum_{i=2}^{m} A^*_{\psi,i} \hat{s}_i = \sum_{i \in [m]} A^*_{\psi,i} ca^{i-1} + d^{-1} \tilde{\lambda}_\psi
$$

where $\tilde{\lambda}_\psi = \sum_{i=2}^m A^*_{\psi,i} \hat{s}_i$ and it is known to $\mathcal{B}$.

To compute $g_1^{\boldsymbol{c}(\tilde{\mathbb{A}}^*, \eta, (1, \boldsymbol{h}); s, \boldsymbol{s})}$ for $\tilde{\mathbb{A}}^*$, $\mathcal{B}$ computes $g_1^{c_1} = g_1^c$ and $g_1^{c_2} = (g_1^c)^{d+\tilde{\nu}} = (g_1^{dc})(g_1^c)^{\tilde{\nu}}$ using the elements in the instance. It randomly selects $\hat{t}_\psi$ and sets $t_\psi = -dcb_\psi + \hat{t}_\psi$. First, it computes $g_1^{c_{\psi,3}} = (g_1^{dcb_\psi})^{-1} g_1^{\hat{t}_\psi}$. The others $g_1^{c_{\psi,1}}$ and $g_1^{c_{\psi,2}}$ can be computed as follows:

- If $\psi \in L_P$, $\mathcal{B}$ computes $g_1^{c_{\psi,1}} = g_1^{\delta\lambda_\psi + \zeta\tilde{t}_\psi}$, $g_1^{c_{\psi,2}} = g_1^{\tilde{t}_\psi(\rho^*(\psi)y_h + y_w)}$ as follows:

$$
\begin{aligned}
g_1^{c_{\psi,1}} &= g_1^{da\sum_{i\in[m]} A^*_{\psi,i} ca^{i-1} + da(d^{-1})\tilde{\lambda}_\psi + (\tilde{\zeta} + \sum_{(j,k)\in L_P\times[m]}(A^*_{j,k}a^k/b_j))(-dcb_\psi + \hat{t}_\psi)} \\
&= g_1^{\boxed{\sum_{i\in[m]} A^*_{\psi,i} dca^i} + a\tilde{\lambda}_\psi + \tilde{\zeta}(-dcb_\psi + \hat{t}_\psi)} \\
&\quad \cdot g_1^{-\boxed{\sum_{(j,k)\in L_P\times[m]} (A^*_{j,k} dca^k b_\psi/b_j)} + \hat{t}_\psi \sum_{(j,k)\in L_P\times[m]} A^*_{j,k} a^k/b_j} \\
&= g_1^{a\tilde{\lambda}_\psi - \tilde{\zeta} dcb_\psi + \tilde{\zeta}\hat{t}_\psi - \boxed{\sum_{\substack{(j,k)\in L_P\times[m]\\ j\neq\psi}} (A^*_{j,k} dca^k b_\psi/b_j)} + \hat{t}_\psi \sum_{(j,k)\in L_P\times[m]} A^*_{j,k} a^k/b_j} \\
&= g_1^{\tilde{\zeta}\hat{t}_\psi} (g_1^a)^{\tilde{\lambda}_\psi} (g_1^{dcb_\psi})^{-\tilde{\zeta}} \prod_{\substack{(j,k)\in L_P\times[m]\\ j\neq\psi}} (g_1^{dca^k b_\psi/b_j})^{-A^*_{j,k}} \prod_{(j,k)\in L_P\times[m]} (g_1^{a^k/b_j})^{A^*_{j,k}\hat{t}_\psi},
\end{aligned}
$$

$$
\begin{aligned}
g_1^{c_{\psi,2}} &= g_1^{\tilde{t}_\psi(\rho^*(\psi)y_h + y_w)} = g_1^{-dcb_\psi(\rho^*(\psi)y_h + y_w) + \hat{t}_\psi(\rho^*(\psi)y_h + y_w)} \\
&= g_1^{-dcb_\psi\left(\rho^*(\psi)\tilde{h} + \boxed{\rho^*(\psi)\sum_{(j,k)\in L_P\times[m]} A^*_{j,k}\cdot a^k/b_j^2} + \tilde{w} - \boxed{\sum_{(j,k)\in L_P\times[m]} \rho^*(j) A^*_{j,k}\cdot a^k/b_j^2}\right)} g_1^{\Phi_6} \\
&= g_1^{-dcb_\psi\left(\rho^*(\psi)\tilde{h} + \tilde{w} - \boxed{\sum_{\substack{(j,k)\in L_P\times[m]\\ j\neq\psi}} (\rho^*(\psi) - \rho^*(j)) A^*_{j,k}\cdot a^k/b_j^2}\right)} g_1^{\Phi_6} \\
&= (g_1^{dcb_\psi})^{-(\rho^*(\psi)\tilde{h} + \tilde{w})} \prod_{(j,k)\in L_P\times[m], j\neq\psi} (g_1^{dca^k b_\psi/b_j^2})^{(\rho^*(\psi) - \rho^*(j))A^*_{j,k}} g_1^{\Phi_6}
\end{aligned}
$$

where $g_1^{\Phi_6} = g_1^{\hat{t}_\psi(\rho^*(\psi)y_h + y_w)}$. $g_1^{\Phi_6}$ can be computed using $g_1^{y_h}$ and $g_1^{y_w}$.

- if $\psi \in L_N$, $\mathcal{B}$ computes $C_{\psi,1} = g_1^{\delta\lambda_\psi + t_\psi\eta y_x}$ as follows:

$$
\begin{aligned}
g_1^{c_{\psi,1}} &= g_1^{da\sum_{i\in[m]} A^*_{\psi,i} ca^{i-1} + da(d^{-1})\tilde{\lambda}_\psi} \\
&\quad \cdot g_1^{(-dcb_\psi + \hat{t}_\psi)(\tilde{x}\sum_{i\in[n_2]} b_i + \sum_{\substack{(i,k)\in[n_2,m], j\in L_N\\ i\neq j}} A^*_{j,k} a^k b_i/b_j^2 + \sum_{(j,k)\in L_N\times[m]} A^*_{j,k} a^k/b_j)} \\
&= g_1^{\boxed{\sum_{i\in[m]} A^*_{\psi,i} dca^i} + a\tilde{\lambda}_\psi - \tilde{x}\sum_{i\in[n_2]} dcb_\psi b_i} \\
&\quad \cdot g_1^{-\sum_{\substack{(i,k)\in[n_2,m], j\in L_N\\ i\neq j}} A^*_{j,k} dca^k b_\psi b_i/b_j^2 - \boxed{\sum_{(j,k)\in L_N\times[m]} A^*_{j,k} dca^k b_\psi/b_j}} \cdot g_1^{\Phi_7} \\
&= g_1^{a\tilde{\lambda}_\psi - \tilde{x}\sum_{j\in[n_2]} dcb_\psi b_j} \\
&\quad \cdot g_1^{-\sum_{\substack{(i,k)\in[n_2,m], j\in L_N\\ i\neq j}} A^*_{j,k} dca^k b_\psi b_i/b_j^2 - \boxed{\sum_{\substack{(j,k)\in L_N\times[m]\\ j\neq\psi}} A^*_{j,k} dca^k b_\psi/b_j}} \cdot g_1^{\Phi_7} \\
&= (g_1^a)^{\tilde{\lambda}_\psi} \prod_{j\in[n_2]} (g_1^{dcb_\psi b_j})^{-\tilde{x}} \prod_{(i,k)\in[n_2,m], j\in L_N, i\neq j} (g_1^{dca^k b_\psi b_i/b_j^2})^{A^*_{j,k}} \\
&\quad \cdot \prod_{(j,k)\in L_N\times[m], j\neq\psi} (g_1^{dca^k b_\psi/b_j})^{A^*_{j,k}} \cdot g_1^{\Phi_7}
\end{aligned}
$$

where $\varPhi_7 = \hat{t}_\psi \eta y_x$. $\mathcal{B}$ can compute $g_1^{\varPhi_7}$ using $g_1^{\eta y_x}$.

Computing $g_1^{c_{\psi,2}}$ is similar to that in the previous case. In this case, y_x and y_y are used instead of y_h and y_w and L_N replaces L_P. $\mathcal{B}$ computes $g_1^{c_{\psi,2}}$ as follows:

$$\begin{aligned} g_1^{c_{\psi,2}} &= g_1^{\tilde{t}_\psi(\rho^*(\psi)y_x+y_y)} = g_1^{-(\rho^*(\psi)\tilde{x}+\tilde{y})dcb_\psi+\sum_{(j,k)\in L_N\times[m],j\neq\psi}\rho^*(j)A^*_{j,k}\cdot dca^k b_\psi/b_j^2} \cdot g_1^{\varPhi_8} \\ &= (g_1^{dcb_\psi})^{-(\rho^*(\psi)\tilde{x}+\tilde{y})} \prod_{(j,k)\in L_N\times[m],j\neq\psi} (g_1^{dca^k b_\psi/b_j^2})^{\rho^*(j)A^*_{j,k}} \cdot g_1^{\varPhi_8} \end{aligned}$$

where $\varPhi_8 = \hat{t}_\psi(\rho^*(\psi)y_x + y_y)$. $g_1^{\varPhi_8}$ can be computed using $g_1^{y_x}$ and $g_1^{y_y}$.

k-type response: When the adversary requests the k-type response to $\mathcal{A}$ for S, $\mathcal{B}$ sets $S' = N(S)$. We let A^*_ψ is the ψth row of the access matrix A^* which is given in the c-type query. By proposition 11 in [13], $\mathcal{B}$ can select a random vector $\boldsymbol{z'} = (1, z'_2, ..., z'_m) \in \mathbb{Z}_p^m$ which satisfies $\langle \boldsymbol{z'}, A_{i'}\rangle = 0$ for all i' such that $\rho(i') \in S'$. Then, $\mathcal{B}$ randomly selects z_1 from $\mathbb{Z}_p$ and sets $\boldsymbol{z} = (z_1, ..., z_m) = z_1\boldsymbol{z'}$.

$\mathcal{B}$ randomly selects $\hat{r}$ from $\mathbb{Z}_p$ and sets $r = \sum_{i\in[m]} z_i a^{n_2+1-i}$ and $r_0 = c\hat{r} - \sum_{i\in[m],i\neq 1} z_i a^{n_2+2-i}$. r_0 is randomly distributed due to $\hat{r}$ which are not used anywhere else. r is randomly distributed due to z_1 which was not used anywhere else.

To compute $g_2^{\boldsymbol{k}(\beta\cdot\alpha,S,\boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})}$ for S. Firstly, $\mathcal{B}$ computes follows:

$$g_2^{d_1} = T^{z_1} \cdot (g_2^{dc})^{\hat{r}} (g_2^c)^{\tilde{\nu}\hat{r}} \prod_{i\in[m],i\neq 1} (g_2^{a^{n_2+2-i}})^{-\tilde{\nu}z_i}$$

If T is equal to $g_2^{da^{n_2+1}}$, D_1 is equal to $g_2^{\delta r+\nu r_0}$ and $\mathcal{B}$ simulates $\mathcal{O}_0^{Sel}(n_2)$ because

$$\begin{aligned} g_2^{d_1} &= g_2^{da(\sum_{i\in[m]} z_i a^{n_2+1-i})} g_2^{(d+\tilde{\nu})(c\hat{r}-\sum_{i\in[m],i\neq 1} z_i a^{n_2+2-i})} \\ &= \boxed{\prod_{i\in[m]} g_2^{z_i d a^{n_2+2-i}}}\boxed{\prod_{i\in[m],i\neq 1} (g_2^{z_i d a^{n_2+2-i}})^{-1}}(g_2^{dc})^{\hat{r}} (g_2^c)^{\tilde{\nu}\hat{r}} \cdot \prod_{i\in[m],i\neq 1} (g_2^{z_i a^{n_2+2-i}})^{-\tilde{\nu}} \\ &= \boxed{(g_2^{da^{n_2+1}})^{z_1}} \cdot (g_2^{dc})^{\hat{r}} (g_2^c)^{\tilde{\nu}\hat{r}} \prod_{i\in[m],i\neq 1} (g_2^{a^{n_2+2-i}})^{-\tilde{\nu}z_i}. \end{aligned}$$

If T is a random from G_2 and we write it as $g_2^{da^{n_2+1}} g_2^{\tilde{\alpha}}$, the randomness is added to $g_2^{d_1}$ and $\mathcal{B}$ simulates $\mathcal{O}_1^{Sel}(n_2)$. Then, it computes the others following:

$$g_2^{d_2} = g_2^r = \prod_{i\in[m]} (g_2^{a^{n_2+1-i}})^{z_i}, \quad g_2^{d_3} = g_2^{r_0} = (g_2^c)^{\hat{r}} \prod_{i\in[m-1]} (g_2^{a^{n_2+1-i}})^{-z_{i+1}}.$$

• **Compute** $g_2^{d_{\psi,1}}$ *and* $g_2^{d_{\psi,2}}$: To compute $g_2^{d_{\psi,1}} = g_2^{-\zeta r + r_\psi(w_\psi y_h + y_w)}$ for all $\psi \in [|S|]$, it computes $g_2^{-\zeta r}$ and $g_2^{r_\psi(w_\psi y_h+y_w)}$, separately. First, it computes

$$\begin{aligned} g_2^{-\zeta r} &= g_2^{-(\tilde{\zeta}+\sum_{(j,k)\in L_P\times[m]} A^*_{j,k}\cdot a^k/b_j)\sum_{i\in[m]} z_i a^{n_2+1-i}} \\ &= g_2^{-\tilde{\zeta}\sum_{i\in[m]} z_i a^{n_2+1-i}} g_2^{-\sum_{\substack{(i,j,k)\in[m]\times L_P\times[m] \\ i\neq k}} A^*_{j,k} z_i \cdot a^{n_2+1+k-i}/b_j} \\ &\cdot g_2^{-\sum_{(i,j)\in[m]\times L_P} A^*_{j,i} z_i \cdot a^{n_2+1}/b_j} \\ &= g_2^{\varPhi_8} \cdot g_2^{\sum_{j\in L_P}\langle A^*_j,\boldsymbol{z}\rangle a^{n_2+1}/b_j} = g_2^{\varPhi_8} \cdot \prod_{j\in L_P, j\notin S} g_2^{\langle A^*_j,\boldsymbol{z}\rangle a^{n_2+1}/b_j} \end{aligned}$$

where $\varPhi_8 = -\tilde{\zeta}\sum_{i\in[m]} z_i a^{n_2+1-i} - \sum_{\substack{(i,j,k)\in[m]\times L_P\times[m] \\ i\neq k}} A^*_{j,k} z_i \cdot a^{n_2+1+k-i}/b_j$. The last equality of the above equation holds because $\langle A^*_j, \boldsymbol{z}\rangle = 0$ for all j such that $\rho^*(j) \in S$. $g_2^{\phi_8}$ can be computed since $\{g_2^{a^i}; \forall i \in [n_2]\}$ and $\{g_2^{a^i/b_j}; \forall (i,j) \in [2n_2, n_2], i \neq n_2+1\}$ are given in the instance.

Secondly, to compute $g_2^{r_\psi(w_\psi y_h + y_w)}$, it creates $\hat{r}_\psi \xleftarrow{R} \mathbb{Z}_p$ and sets $r_\psi = \hat{r}_\psi - \sum_{\substack{(i,i') \in [m] \times L_P \\ \rho^*(i') \notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{w_\psi - \rho^*(i')}$ where $w_\psi \neq \rho^*(i')$ due to $w_\psi \in S$. $\mathcal{B}$ computes

$$
\begin{aligned}
g_2^{r_\psi(w_\psi y_h + y_w)} &= g_2^{\hat{r}_\psi(w_\psi y_h + y_w) - \sum_{\substack{(i,i') \in [m] \times L_P \\ \rho^*(i') \notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{w_\psi - \rho^*(i')} (\tilde{h} w_\psi + \tilde{w})} \\
&\quad \cdot g_2^{- \sum_{\substack{(i,i') \in [m] \times L_P \\ \rho^*(i') \notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{w_\psi - \rho^*(i')} \sum_{(j,k) \in L_P \times [m]} (w_\psi - \rho^*(j)) A^*_{j,k} a^k / b_j^2)} \\
&= g_2^{\Phi_9} \cdot g_2^{- \sum_{\substack{(i,i',j,k) \in [m] \times L_P^2 \times [m] \\ \rho^*(i') \notin S}} \frac{w_\psi - \rho^*(j)}{w_\psi - \rho^*(i')} A^*_{j,k} z_i a^{n_2+1+k-i} b_{i'} / b_j^2} \\
&= g_2^{\Phi_9} \cdot g_2^{- \sum_{\substack{(i,i',j,k) \in [m] \times L_P^2 \times [m] \\ \rho^*(i') \notin S, (j \neq i') \vee (i \neq k)}} \frac{w_\psi - \rho^*(j)}{w_\psi - \rho^*(i')} A^*_{j,k} z_i a^{n_2+1+k-i} b_{i'} / b_j^2} \\
&\quad \cdot g_2^{- \sum_{\substack{(i,i',j,k) \in [m] \times L_P^2 \times [m] \\ \rho^*(i') \notin S, (j = i') \wedge (i = k)}} \frac{w_\psi - \rho^*(j)}{w_\psi - \rho^*(i')} A^*_{j,k} z_i a^{n_2+1+k-i} b_{i'} / b_j^2} \\
&= g_2^{\Phi_9} \cdot g_2^{\Phi_{10}} \cdot g_2^{- \sum_{j \in L_P, \rho^*(j) \notin S} \langle A^*_j, \boldsymbol{z} \rangle a^{n_2+1} / b_j}
\end{aligned}
$$

where $\Phi_9 = \hat{r}_\psi(w_\psi y_h + y_w) - \sum_{\substack{(i,i') \in [m] \times L_P \\ \rho^*(i') \notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{w_\psi - \rho^*(i')} (\tilde{h} w_\psi + \tilde{w})$ and $\Phi_{10} = - \sum_{\substack{(i,i',j,k) \in [m] \times L_P^2 \times [m] \\ \rho^*(i') \in S, (j \neq i') \vee (i \neq k)}} \frac{w_\psi - \rho^*(j)}{w_\psi - \rho^*(i')} A^*_{j,k} z_i a^{n_2+1+k-i}$
Both $g_2^{\Phi_9}$ and $g_2^{\Phi_{10}}$ can be computed since $g_2^{y_h}$ and $g_2^{y_w}$ can be computed and $\{g_2^{a^i b_j}; \forall (i,j) \in [n_2, n_2]\}$ and $\{g_2^{a^i b_j / b_{j'}^2}; \forall (i,j,j') \in [2n_2, n_2, n_2], j \neq j'\}$ are given in the instance.

Because $g_2^{- \sum_{j \in L_P, \rho^*(j) \notin S} \langle A^*_j, \boldsymbol{z} \rangle a^{n_2+1} / b_j}$ is cancelled out when $g_2^{-\zeta r}$ and $g_2^{r_\psi(w_\psi y_h + y_w)}$ are multiplied, $\mathcal{B}$ can compute $g_2^{d_{\psi,1}} = g_2^{-\zeta r} g_2^{r_\psi(w_\psi y_h + y_w)}$ and $g_2^{d_{\psi,2}} = g_2^{r_\psi}$

$$
g_2^{d_{\psi,1}} = g_2^{\Phi_8} g_2^{\Phi_9} g_2^{\Phi_{10}}, \quad g_2^{d_{\psi,2}} = g_2^{\hat{r}_\psi} \prod_{\substack{(i,i') \in [m] \times L_P \\ \rho^*(i') \notin S}} (g_2^{b_{i'} a^{n_2+1-i}})^{\frac{z_i}{w_\psi - \rho^*(i')}}.
$$

• **Compute** $g_2^{d'_{\psi,1}}$ *and* $g_2^{d'_{\psi,2}}$: In order to compute $g_2^{d'_{\psi,1}} = (g_2^{\eta w_\psi y_x + \eta y_y})^{r'_\psi}$ and $g_2^{d'_{\psi,2}} = (g_2^{\eta})^{r'_\psi}$ such that $\sum_{\psi \in |S|} r'_\psi = r$, B first sets $g_2^{d'_{\psi,1}} \leftarrow 1_{G_2}$ and $g_2^{d'_{\psi,2}} \leftarrow 1_{G_2}$. Then, it updates $g_2^{d'_{\psi,1}}$ and $g_2^{d'_{\psi,2}}$ by multiplying $g_2^{b_\pi r(w_\psi y_x + y_y)}$ and $g_2^{b_\pi r}$ for all $\pi \in [n_2]$. This process allows $\mathcal{B}$ to compute $\prod_{\pi \in [n_2]} g_2^{b_\pi r(w_\psi y_x + y_y)} = g_2^{\eta r(w_\psi y_x + y_y)}$ and $\prod_{\pi \in [n_2]} g_2^{b_\pi r} = g_2^{\eta r}$. After computing these values, $\mathcal{B}$ will re-randomize the values so that they are properly distributed. The following **Update** process is repeated for all $\pi = 1, ..., n_2$.

Update: $\mathcal{B}$ can compute $g_2^{b_\pi r}$ because $g_2^{b_\pi r} = \prod_{i \in [m]} (g_2^{a^{n_2+1-i} b_\pi})^{-z_i}$ and $g_2^{a^i b_j}$ is given in the instance. For all $\pi \in [n_2]$, it then computes

$$
g_2^{b_\pi r(w_\psi y_x + y_y)} = g_2^{\sum_{i \in [m]} z_i a^{n_2+1-i} b_\pi (\rho(\pi) y_x + y_y)}
$$

There are three cases to compute $g_2^{b_\pi r(w_\psi y_x + y_y)}$ as follows:

[Case 1] If $\pi \in L_N \wedge \rho(\pi) = x'_\psi \wedge x_\psi = w_\psi \in S$,

$$
\begin{aligned}
g_2^{b_\pi r(w_\psi y_x + y_y)} &= g_2^{\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi \left(\rho(\pi)\tilde{x} + \tilde{y} + \boxed{\rho(\pi) \sum_{(j,k)\in L_N\times[m]} a^k A^*_{j,k}/b_j^2}\right)} \\
&\quad \cdot g_2^{-\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi \left(\boxed{\sum_{(j,k)\in L_N\times[m]} \rho^*(j) A^*_{j,k} a^k / b_j^2}\right)} \\
&= g_2^{\Phi_{11}} \cdot g_2^{\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi \boxed{\sum_{\substack{(j,k)\in L_N\times[m] \\ j\neq\pi}} (\rho(\pi) - \rho^*(j)) A^*_{j,k} a^k / b_j^2}} \\
&= g_2^{\Phi_{11}} \prod_{(i,j,k)\in[m]\times L_N\times[m], j\neq\pi} \left(g_2^{a^{n_2+1+k-i} b_\pi / b_j^2}\right)^{(\rho(\pi)-\rho^*(j)) A^*_{j,k} z_i}
\end{aligned}
$$

where $\Phi_{11} = \prod_{i\in[m]} z_i a^{n_2+1-i} b_\pi \big(\rho(\pi)\tilde{x} + \tilde{y}\big)$. $\mathcal{B}$ can compute $g_2^{\Phi_{11}}$ since $\{g_2^{a^i b_j};\ \forall (i,j) \in [n_2, n_2]\}$ is given in the instance. a^k/b_π^2 was cancelled out in the boxed terms since $\pi \in L_N$. In the above equation, $g_2^{a^{n_2+1}/b_\pi}$ which is not given in the instance does not appear since $j \neq \pi$. Therefore, $g_2^{d'_{\pi,1}}$ can be computed.

[Case 2] If $\pi \in L_N \wedge \rho(\pi) = x'_\psi \wedge x_\psi \notin S$,

$$
\begin{aligned}
g_2^{b_\pi r(w_\psi y_x + y_y)} &= g_2^{\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi \left(w_\psi \tilde{x} + \tilde{y} + \sum_{(j,k)\in L_N\times[m]} (w_\psi - \rho^*(j)) a^k A^*_{j,k}/b_j^2\right)} \\
&= g_2^{\Phi_{12}} \cdot g_2^{\sum_{(i,j,k)\in[m]\times L_N\times[m]} (w_\psi - \rho^*(j)) A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi / b_j^2} \\
&= g_2^{\Phi_{12}} \cdot g_2^{\sum_{\substack{(i,j,k)\in[m]\times L_N\times[m] \\ (k\neq i)\vee(j\neq\pi)}} (w_\psi - \rho^*(j)) A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi / b_j^2} \\
&\quad \cdot g_2^{\sum_{\substack{(i,j,k)\in[m]\times L_N\times[m] \\ (k=i)\wedge(j=\pi)}} (w_\psi - \rho^*(j)) A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi / b_j^2} \\
&= g_2^{\Phi_{12}} \cdot g_2^{\Phi_{13}} \cdot g_2^{\sum_{i\in[m]} (w_\psi - \rho^*(j)) A^*_{\pi,i} z_i a^{n_2+1}/b_\pi} = g_2^{\Phi_{12}} \cdot g_2^{\Phi_{13}}
\end{aligned}
$$

where $\Phi_{12} = \sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi \big(w_\pi \tilde{x} + \tilde{y}\big)$ and

$$
\Phi_{13} = \sum_{\substack{(i,j,k)\in[m]\times L_N\times[m] \\ (k\neq i)\vee(j\neq\pi)}} (w_\psi - \rho^*(j)) A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi / b_j^2.
$$

The last equality of the above equation holds since $\langle A^*_\pi, \boldsymbol{z}\rangle = 0$ for all π such that $\rho^*(\pi) \in S'$ (i.e. $\rho^*(\pi) \notin S$). $g_2^{\Phi_{12}}$ and $g_2^{\Phi_{13}}$ can be computed since $\{g_2^{a^i b_j}; \forall (i,j) \in [n_2, n_2]\}$ and $\{g_2^{a^i b_j / b_{j'}^2}; \forall (i,j,j') \in [2n_2, n_2, n_2], j \neq j'\}$ are given.

[Case 3] If $\pi \notin L_N$,

$$
\begin{aligned}
g_2^{b_\pi r(w_\psi y_x + y_y)} &= g_2^{\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi \left(w_\psi \tilde{x} + \tilde{y} + \sum_{(j,k)\in L_N\times[m]} (w_\psi - \rho^*(j)) a^k A^*_{j,k}/b_j^2\right)} \\
&= g_2^{\Phi_{12}} \cdot g_2^{\sum_{(i,j,k)\in[m]\times L_N\times[m]} (w_\psi - \rho^*(j)) A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi / b_j^2} \\
&= g_2^{\Phi_{12}} \prod_{(i,j,k)\in[m]\times L_N\times[m]} \left(g_2^{a^{n_2+1+k-i} b_\pi / b_j^2}\right)^{(w_\psi - \rho^*(j)) A^*_{j,k} z_i}
\end{aligned}
$$

where Φ_{12} is identical to that in the previous case. It should be noted that $\pi \notin L_N$. Therefore, $g_2^{b_\pi r'(w_i y_x + y_y)}$ can be calculated since all terms are given in the instance. In other words, $g_2^{a^{n_2+1+k-i}/b_\pi}$ which was not given in the instance does not appear in the above equation because $j \in L_N$.

Now, it updates $g_2^{d'_{\psi,1}} \leftarrow g_2^{d'_{\psi,1}} \cdot (g_2^{k'_{\pi,1}})^{1/|S|}$ and $g_2^{d'_{\psi,2}} \leftarrow g_2^{d'_{\psi,2}} \cdot (g_2^{k'_{\pi,2}})^{1/|S|}$.

Re-randomization: After the above updating process, $\mathcal{B}$ can derive

$$g_2^{d'_{\psi,1}} = (g_2^{r(w_\psi y_x + y_y)})^{\sum_{\pi \in [n_2]} b_\pi r/|S|} = (g_2^{r(w_\psi y_x + y_y)})^{\eta r/|S|},$$

$$g_2^{d'_{\psi,2}} = g_2^{\sum_{\pi \in [n_2]} b_\pi r/|S|} = g_2^{\eta r/|S|}.$$

This sets $r'_\psi = r/|S|$ for all $\psi \in [|S|]$. Therefore, we need to re-randomize those $g_2^{d'_{\psi,1}}$ and $g_2^{d'_{\psi,2}}$. In order to re-randomize them, we randomly select $\hat{r}'_1, ..., \hat{r}'_{|S|}$ such that $\hat{r}'_1 + ... + \hat{r}'_{|S|} = 0$ and sets $g_2^{d'_{\psi,1}} \leftarrow g_2^{d'_{\psi,1}} \cdot (g_2^{r(w_\psi y_x + y_y)})^{\hat{r}'_\psi}$ and $g_2^{d'_{\psi,2}} \leftarrow g_2^{d'_{\psi,2}} \cdot g_2^{\hat{r}'_\psi}$. This implicitly sets $r'_\psi = r/|S| + \hat{r}'_\psi/b$. Due to $\hat{r}'_\psi$, r'_ψ is randomly distributed. Moreover, $\sum_{\psi \in [|S|]} r'_\psi = |S| \cdot r/|S| + \sum_{i \in [|S|]} \hat{r}'_\psi/b = r$. □

Lemma 9 *Suppose there exists a PPT adversary $\mathcal{A}$ who can distinguish between $\hat{\mathcal{O}}_0^{Cos}(n_1)$ and $\hat{\mathcal{O}}_1^{Cos}(n_1)$ with non-negligible advantage ϵ. Then, we can build an algorithm $\mathcal{B}$ breaking (Asymmetric) n_1-DBDHE assumption with ϵ using $\mathcal{A}$ with an attributes set of size $k < n_1$.*

Proof: Given $D := \{g_1, g_2, g_1^c, g_2^c\} \cup \{g_1^{z_1}, g_2^{z_2} | z_1 \in Z_1, z_2 \in Z_2\}$ and T_β from (Asymmetric) n_1-DBDHE assumption where $Z_1 = Z_2 = \{dc, b^1, ..., b^{n_1}, b^{n_1+2}, ..., b^{2n_1}\}$. $\mathcal{B}$ will simulate either $\hat{\mathcal{O}}_0^{Cos}(n_1)$ or $\hat{\mathcal{O}}_1^{Cos}(n_1)$ using $\mathcal{A}$.

Initial response When $\mathcal{A}$ sends the initial query to $\mathcal{B}$, $\mathcal{B}$ randomly selects η from $\mathbb{Z}_p$. It outputs $\{g_1, g_2, g_1^\eta, g_2^\eta\}$.

k-type response When $\mathcal{A}$ sends the k-type query for a set of attributes $S^* = (w_1^*, ..., w_k^*)$ where $k < n_1$ to $\mathcal{B}$. $\mathcal{B}$ computes a_i which is the coefficient of z^{i-1} in $P(z) = \prod_{y \in S^*}(y - z)$ and set $a_{k+2} = ... = a_{n_1+1} = 0$. It then randomly selects $\tilde{\zeta}$, ν, $\{\tilde{y}_i, \tilde{y}'_i; \forall i \in [n_1]\}$ from $\mathbb{Z}_p$. It implicitly sets $\delta = \delta_0 \cdot b^{n_1+1}/c$ and $\zeta = \tilde{\zeta} - \sum_{i \in n_1} a_i \cdot b^i$. For all $i \in [n_1]$, it also sets

$$y_i = \tilde{y}_i + b^i,\ y'_i = \tilde{y}'_i + \sum_{j \in [k]} (w_j^*)^{i-1} b^{n_1+1-j},\ \eta y'_i = \eta \tilde{y}'_i + \eta \sum_{j \in [k]} (w_j^*)^{i-1} b^{n_1+1-j}.$$

$\mathcal{B}$ selects $r_0 \xleftarrow{R} \mathbb{Z}_p$ and sets $r_1 = -dc + \tilde{r}_1$ and $r_2 = dc$ where $\tilde{r}_1$ is randomly generated from $\mathbb{Z}_p$. Allocating dc to r_2 is hidden to $\mathcal{A}$ since d appears nowhere else except r_1. In r_1, due to $\tilde{r}_1$, the value dc are not revealed. Hence, both r_1 and r_2 are properly distributed to the adversary. To compute $g_2^{\boldsymbol{k}(\beta \cdot \alpha, S^*, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); \boldsymbol{r})}$, $\mathcal{B}$ sets

$$g_2^{d_1} = T^{\delta_0} \cdot g_2^{\nu r_0}, \quad g_2^{d_2} = g_2^{r_0}, \quad g_2^{d_3} = g_2^{dc}.$$

If $T = g_2^{db^{n_1+1}}$, then $g_2^{d_1} = g_2^{\delta r_2 + \nu r_0}$. Therefore, $\mathcal{B}$ simulates the k-type response of $\hat{\mathcal{O}}_0^{Cos}(n_1)$. If T is random from G_2 and we denote it $T = g_2^{db^{n_1+1}} g_2^\alpha$, $g_2^{d_1} = g_2^{\alpha + \delta r_2 + \nu r_0}$, $\mathcal{B}$ simulates the k-type outputs of $\hat{\mathcal{O}}_1^{Cos}(n_1)$.

- **Compute** Using $r_1 = -dc + \tilde{r}_1$, $\mathcal{B}$ computes $g_2^{d_4}$ and $g_2^{d_5}$ as follows:

$$\begin{aligned} g_2^{d_4} &= g_2^{-\zeta r_2 + r_1(y_1 a_1 + ... + y_{n_1} a_{n_1})} \\ &= g_2^{-(\tilde{\zeta} - \sum_{i \in n_1} a_i \cdot b^i) dc + (-dc + \tilde{r}_1)(\sum_{i \in n_1} a_i(\tilde{y}_i + b^i))} \\ &= g_2^{-\tilde{\zeta} dc + \tilde{r}_1(\sum_{i \in n_1} a_i(\tilde{y}_i + b^i))} \\ g^{d_5} &= (g_2^{dc})^{-1} g_2^{\tilde{r}_1}. \end{aligned}$$

- **Compute** $g_2^{d'_6}$ and $g_2^{d'_7}$: $\mathcal{B}$ computes $g_2^{d'_6}$ and $g_2^{d'_7}$ as follows:

$$g_2^{d'_6} = g_2^{r_2 \eta (\sum_{i \in [n_1]} a_i y'_i)} = g_2^{\eta dc \sum_{i \in [n_1]} a_i (\tilde{y}'_i + \sum_{j \in [k]} (w_j^*)^{i-1} b^{n_1+1-j})} = g_2^{\eta dc \sum_{i \in [n_1]} a_i \tilde{y}'_i}$$

$$g_2^{d_7'} = g_2^{dc}$$

In $g_2^{d_6'}$, $\sum_{(i,j)\in[n_1,k]} a_i (w_j^*)^{i-1} b^{n_1+1-j} = 0$ since $w_j^* \in S^*$. Finally, it outputs $g_2^{\boldsymbol{k}(\beta\cdot\alpha, S^*, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); \boldsymbol{r})}$.

c-type response When $\mathcal{A}$ sends c-type query for $\tilde{\mathbb{A}}$ to $\mathcal{B}$ after it sent the k-type query. Since S^* of k-type query does not satisfy $\tilde{\mathbb{A}} = NM(\mathbb{A})$, there exists $S' = N(S^*) \notin \mathbb{A}$ where $\mathbb{A} = (A, \rho)$ and A is an $\ell \times m$ matrix. We let A_ψ is the ψth row of A. By proposition 11 in [13], $\mathcal{B}$ can select a random vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_m) \in \mathbb{Z}_p^m$ which satisfies $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$ for all i' such that $\rho(i') \in S'$ and $\langle (1,0,...0), \boldsymbol{\theta}\rangle = 1$. It then implicitly sets $\boldsymbol{\phi} = (s, s_2, ..., s_m) = c \cdot \boldsymbol{\theta} + (\delta)^{-1}\boldsymbol{\mu}$ where $\boldsymbol{\mu} = (0, \mu_2, ..., \mu_m)$ is randomly selected from $\mathbb{Z}_p^m$. This implicitly sets $s = c$. c was already used in ζ, r_1 and r_2, but its value was not revealed because of $\tilde{\zeta}$, $\tilde{r}_1$ and d, respectively. Therefore, assigning c to s is hidden to the adversary. The other values $s_2, ..., s_m$ are randomly distributed due to $\boldsymbol{\mu}$.

To compute the c-type response $g_1^{\boldsymbol{c}(\tilde{\mathbb{A}}, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h}); s, \boldsymbol{s})}$, it sets $g_1^{c_1} = (g_1^c)$ and $g_1^{c_2} = (g_1^c)^\nu$. To compute the other elements, it also computes $g_1^{c_{i',1}}, g_1^{c_{i',2}}, ..., g_1^{c_{i',N+2}}$ for $i' \in [\ell]$ using one of following four cases.

[**Case 1**] $\rho(i')$ is not a negated attribute and $\rho(i') \in S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$).

For all $j \in [N-1]$, $\mathcal{B}$ randomly selects $t_{i'} \in \mathbb{Z}_p$ and computes

$$g_1^{c_{i',1}} = g_1^{\langle A_{i'}, \boldsymbol{\mu}\rangle + \zeta t_{i'}} = g_1^{\langle A_{i'}, \boldsymbol{\mu}\rangle + \tilde{\zeta} t_{i'}} \prod_{i\in[n_1]} (g_1^{b^i})^{a_i t_{i'}}, \quad g_1^{c_{i',2}} = {g_1}^{t_{i'}},$$

$$g_1^{c_{i',2+j}} = g_1^{-t_{i'}(\tilde{y}_{j+1} - \rho(i')^j \tilde{y}_1 + b^{j+1} - \rho(i')^j b)} = g_1^{-t_{i'}(\tilde{y}_{j+1} - \rho(i')^j \tilde{y}_1)} g_1^{b^{j+1}} (g_1^b)^{-\rho(i')^j}$$

[**Case 2**] $\rho(i')$ is not a negated attribute and $\rho(i') \notin S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle \neq 0$). First, $\mathcal{B}$ can compute $\delta\lambda_i = v_1\delta_0 b^{n+1} + v_2$ with known values v_1, v_2 and δ_0. It sets $t_{i'} = \tilde{t}_{i'} - v_1\delta_0(\sum_{i\in[n_1]} \rho(i')^{n-i} b^i)/A_1$ where $A_1 = (\sum_{i\in[n_1]} a_i \rho(i')^{i-1})$.

$$\begin{aligned} g_1^{c_{i',1}} &= g_1^{v_1\delta_0 b^{n_1+1} + v_2 + \tilde{\zeta}\tilde{t}_{i'} - \tilde{t}_i \sum_{i\in n_1} b^i a_i - \tilde{\zeta} v_1 \delta_0 (\sum_{i\in[n_1]} \rho(i')^{n_1-i} b^j)/A_1} \\ &\quad \cdot g_1^{-v_1\delta_0(\sum_{i,j\in[n_1]^2} a_i \rho(i')^{n_1-j} b^{i+j})/A_1)} \\ &= g_1^{v_2 + \tilde{\zeta}\tilde{t}_{i'} + \tilde{t}_i \sum_{i\in n_1} b^i a_i - \tilde{\zeta} v_1\delta_0(\sum_{i\in[n_1]} \rho(i')^{n_1-i} b^j)/A_1} \\ &\quad \cdot g_1^{-v_1\delta_0(\sum_{i,j\in[n_1]^2, i+j\neq n_1+1} a_i \rho(i')^{n_1-j} b^{i+j})/A_1)}, \end{aligned}$$

$$\begin{aligned} g_1^{c_{i',2}} &= {g_1}^{\tilde{t}_{i'}} \prod_{i\in n_1} (g_1^{b^i})^{-v_1\delta_0\rho(i')^{n-i}/A_1}, \\ g_1^{c_{i',2+j}} &= g_1^{-\tilde{t}_{i'}(\tilde{y}_{j+1} - \rho(i')^j \tilde{y}_1 + b^{j+1} - \rho(i')^j b)} \\ &\quad \cdot g_1^{v_1\delta_0(\sum_{i\in[n_1]} \rho(i')^{n-i} b^i)(\tilde{y}_{j+1} - \rho(i')^j \tilde{y}_1 + b^{j+1} - \rho(i')^j b)/A_1} \\ &= (g_1^{c_{i',2+j}})^{(\tilde{y}_{j+1} - \rho(i')^j \tilde{y}_1)} g_1^{-\tilde{t}_{i'}(b^{j+1} - \rho(i')^j b)} \\ &\quad \cdot g_1^{v_1\delta_0(\sum_{i\in[n_1]} \rho(i')^{n-i} b^{i+j+1})/A_1} g_1^{-v_1\delta_0(\sum_{i\in[n_1]} \rho(i')^{n+j-i} b^{i+1})/A_1} \\ &= (g_1^{c_{i',2+j}})^{(\tilde{y}_{j+1} - \rho(i')^j \tilde{y}_1)} g_1^{-\tilde{t}_{i'}(b^{j+1} - \rho(i')^j b)} \\ &\quad \cdot g_1^{v_1\delta_0(\sum_{i\in[n_1]\setminus\{n_1-j\}} \rho(i')^{n-i} b^{i+j+1})/A_1} g_1^{-v_1\delta_0(\sum_{i\in[n_1-1]} \rho(i')^{n_1+j-i} b^{i+1})/A_1} \end{aligned}$$

[**Case 3**] $\rho(i')$ is a negated attribute and $\rho(i')' \in S'$ (i.e. $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$).

$\mathcal{B}$ randomly selects $t_{i'}$ from $\mathbb{Z}_p$. Because $\langle A_{i'}, \boldsymbol{\theta}\rangle = 0$, $\mathcal{B}$ can compute

$$g_1^{c_{i',1}} = g_1^{\langle A_{i'}, \boldsymbol{\mu}\rangle - \eta y_1' t_{i'}} = g_1^{\langle A_{i'}, \boldsymbol{\mu}\rangle - \eta t_{i'} \tilde{y}_1'} \prod_{j \in [k]} (g_1^{b^{n_1+1-j}})^{\eta t_{i'}} \qquad g_1^{c_{i',2}} = g_1^{t_{i'}}$$

$$g_1^{c_{i',2+j}} = g_1^{-t_{i'}(\tilde{y}_{j+1}' - \rho(i')^j \tilde{y}_1')} \prod_{j' \in [k]} (g_1^{b^{n_1+1-j'}})^{((w_{j'}^*)^i - \rho(i')^i)}$$

[**Case 4**] $\rho(i')$ is a negated attribute and $\rho(i')' \notin S'$ (i.e. $\rho(i') \in S$ and $\langle A_{i'}, \boldsymbol{\theta}\rangle \neq 0$).

First, $\mathcal{B}$ can compute $\delta\lambda_i = v_1\delta_0 b^{n+1} + v_2$ with known values v_1, v_2 and δ_0. It sets $t_{i'} = \tilde{t}_{i'} + v_1\delta_0 b^{i'})/\eta$.

$$\begin{aligned}
g_1^{c_{i',1}} &= g_1^{v_1\delta_0 b^{n_1+1} + v_2 + \eta\tilde{y}_1' + \eta\sum_{i\in[k]} b^{n_1+1-i} + \eta\tilde{y}_1' v_1\delta_0 b^{i'}/\eta - v_1\delta_0 \sum_{i\in[k]} b^{n_1+1-i+i'}} \\
&= g_1^{v_2 + \eta\tilde{y}_1' + \eta\sum_{i\in[n_1]} b^{n_1+1-i} + \eta\tilde{y}_1' v_1\delta_0 b^{i'}/\eta - v_1\delta_0 \sum_{i\in[n_1]\setminus i'} b^{n_1+1-i+i'}} \\
g_1^{c_{i',2}} &= g_1^{\tilde{t}_{i'}} (g_1^{b^{i'}})^{v_1\delta_0/\eta}, \\
g_1^{c_{i',2+j}} &= g_1^{-\tilde{t}_{i'}(\tilde{y}_{j+1}' - \rho(i')^j \tilde{y}_1' + \sum_{i\in[k]} (w_i^*)^j b^{n_1+1-i} - \rho(i')^j \sum_{i\in[k]} b^{n_1+1-i})} \\
&\quad \cdot g_1^{-v_1\delta_0 b^{i'}/\eta)(\tilde{y}_{j+1}' - \rho(i')^j \tilde{y}_1' + \sum_{i\in[k]} (w_i^*)^j b^{n_1+1-i} - \rho(i')^j \sum_{i\in[k]} b^{n_1+1-i})} \\
&= (g_1^{c_{i',2+j}})^{(\tilde{y}_{j+1}' - \rho(i')^j \tilde{y}_1')} g_1^{-\tilde{t}_{i'}(\sum_{i\in[k]} ((w_i^*)^j - \rho(i')^j) b^{n_1+1-i})} \\
&\quad \cdot g_1^{-v_1\delta_0 b^{i'}(\sum_{i\in[k]} ((w_i^*)^j - \rho(i')^j) b^{n_1+1-i})/\eta} \\
&= (g_1^{c_{i',2+j}})^{(\tilde{y}_{j+1}' - \rho(i')^j \tilde{y}_1')} g_1^{-\tilde{t}_{i'}(\sum_{i\in[k]} ((w_i^*)^j - \rho(i')^j) b^{n_1+1-i})} \\
&\quad \cdot g_1^{-v_1\delta_0 (\sum_{i\in[k]\setminus\{i'\}} ((w_i^*)^j - \rho(i')^j) b^{n_1+1-i+i'})/\eta}
\end{aligned}$$

Therefore, if $T = g_2^{db^{n_1+1}}$, $\mathcal{B}$ simulates $\hat{\mathcal{O}}_0^{Cos}(n_1)$. If T is a random from G_2, $\mathcal{B}$ simulates $\hat{\mathcal{O}}_1^{Cos}(n_1)$. $\square$

Lemma 10. *Suppose there exists an $\mathcal{A}$ who can distinguish between two oracles $\hat{\mathcal{O}}_0^{Sel}(n_2)$ and $\hat{\mathcal{O}}_1^{Sel}(n_2)$ with a non-negligible advantage ϵ. Then, we can build $\mathcal{B}$ breaking A2-(n_2) with ϵ using $\mathcal{A}$ with an access matrix of size $\ell \times m$ where $\ell, m \leq n_2$.*

Proof: Given $D := \{g_1, g_2, g_1^c, g_2^c\} \cup \{g_1^{z_1}, g_2^{z_2} | z_1 \in Z_1, z_2 \in Z_2\}$ and T_β where

$$\begin{aligned}
Z_1 = \{ \quad & \forall(i,j) \in [n_2, n_2],\ dc, a, b_j, dcb_j, dcb_ib_j, a^i/b_j^2 \\
& \forall(i,j,j') \in [2n_2, n_2, n_2], j \neq j',\ a^ib_j/b_{j'}^2 \\
& \forall(i,j,j') \in [n_2, n_2, n_2], j \neq j',\ dca^ib_j/b_{j'}, dca^ib_j/b_{j'}^2 \\
& \forall(i,j,j',j'') \in [n_2, n_2, n_2, n_2], j \neq j', j' \neq j''\},\ dca^ib_jb_{j'}/b_{j''}^2 \quad \},
\end{aligned}$$

$$\begin{aligned}
Z_2 = \{ \quad & \forall(i,j) \in [n_2, n_2],\ dc, a^i, a^ib_j, a^i/b_j^2 \\
& \forall(i,j) \in [2n_2, n_2], i \neq n_2+1,\ a^i/b_j \\
& \forall(i,j,j') \in [2n_2, n_2, n_2], j \neq j',\ a^ib_j/b_{j'}^2 \quad \}
\end{aligned}$$

from A2-(n_2), $\mathcal{B}$ will simulate either $\hat{\mathcal{O}}_0^{Sel}(n_2)$ or $\hat{\mathcal{O}}_1^{Sel}(n_2)$ using $\mathcal{A}$.

Initial response: When the adversary $\mathcal{A}$ requests an initial instance to $\mathcal{B}$. $\mathcal{B}$ computes $g_1^\eta = g_2^{\sum_{i\in[n_2]} b_i}$ and $g_2^\eta = g_2^{\sum_{i\in[n_2]} b_i}$ and outputs $\{g_1, g_2, g_1^\eta, g_2^\eta\}$.

c-type response: When $\mathcal{A}$ sends the c-type query for an access policy $\tilde{\mathbb{A}}^* = NM(\mathbb{A}^*)$ where $\mathbb{A}^* = (A^*, \rho^*)$ and the access matrix A^* is an $\ell \times m$ matrix where $\ell, m \leq n_2$. We define two sets

$L' = \{i | i \in [\ell] \wedge \rho^*(i) = x'_i\}$ for negated attributes and $L = \{i | i \in [\ell] \cap \rho^*(i) = x_i\}$ for non-negated attributes. $\mathcal{B}$ randomly selects $\tilde{h}_1, ..., \tilde{h}_N$, $\tilde{h}'_1, ..., \tilde{h}'_N$, $\tilde{w}$, $\tilde{x}$, $\tilde{y}, \tilde{\zeta}$, $\tilde{\nu}$ from $\mathbb{Z}_p$ and implictly sets

$$\delta = da, \quad \zeta = \tilde{\zeta} + \sum_{(j,k) \in L \times [m]} A^*_{j,k} \cdot a^k / b_j, \quad \nu = d + \tilde{\nu},$$

For all $i \in [N]$,

$$y_i = \tilde{h}_i + \sum_{(j,k) \in L \times [m]} \rho(j)^{i-1} A^*_{j,k} a^k / b_j^2, \ y'_i = \tilde{h}'_i + \sum_{(j,k) \in L' \times [m]} \rho(j)^{i-1} A^*_{j,k} a^k / b_j^2$$

Since $g_1, g_1^{b_j}, g_1^{a^k/b_j}, g_1^{a^k/b_j^2}$ and $g_1^{a^k b_i / b_j^2}$ are given in the instance, $\mathcal{B}$ can compute

$$g_1^{y_i} = g_1^{\tilde{h}_i} \prod_{(j,k) \in L \times [m]} (g_1^{a^k/b_j^2})^{\rho(j)^{i-1} A^*_{j,k}}, \ g_1^{y'_i} = g_1^{\tilde{h}'_i} \prod_{(j,k) \in L' \times [m]} (g_1^{a^k/b_j^2})^{\rho^*(j)^{i-1} A^*_{j,k}},$$

$$\begin{aligned} g_1^{\eta y'_1} &= (\prod_{i \in [n_2]} g_1^{b_i})^{\tilde{h}'_1} \prod_{(i,j,k) \in [n_2] \times L' \times [m]} (g_1^{a^k b_i / b_j^2})^{A^*_{j,k}} \\ &= (\prod_{i \in [n_2]} g_1^{b_i})^{\tilde{h}'_i} \prod_{\substack{(i,j,k) \in [n_2] \times L' \times [m] \\ i \neq j}} (g_1^{a^k b_i / b_j^2})^{A^*_{j,k}} \prod_{(j,k) \in L' \times [m]} (g_1^{a^k / b_j})^{A^*_{j,k}}. \end{aligned}$$

For a negated attribute w'_i, we define $g_1^{w'_i} := g_1^{w_i}$ and $g_2^{w'_i} := g_2^{w_i}$ where $g_1 \in G_1$ and $g_2 \in G_2$. $\mathcal{B}$ randomly selects $\hat{s}_2, ..., \hat{s}_m$ from $\mathbb{Z}_p$ and implictly sets $\boldsymbol{\phi} = c(1, a, a^2, ..., a^{m-1}) + d^{-1}(0, \hat{s}_2, \hat{s}_3, ..., \hat{s}_m)$. This sets $s = c$. Due to $\hat{s}_2, \hat{s}_3, ..., \hat{s}_m$, s does not correlate to the other coordinates in $\boldsymbol{\phi}$. For A^*_ψ where $\psi \in [\ell]$, $\mathcal{B}$ sets

$$\lambda_\psi = \langle A^*_\psi, \boldsymbol{\phi} \rangle = \sum_{i \in [m]} A^*_{\psi,i} c a^{i-1} + d^{-1} \sum_{i=2}^{m} A^*_{\psi,i} \hat{s}_i = \sum_{i \in [m]} A^*_{\psi,i} c a^{i-1} + d^{-1} \tilde{\lambda}_\psi$$

where $\tilde{\lambda}_\psi = \sum_{i=2}^m A^*_{\psi,i} \hat{s}_i$ and it is known to $\mathcal{B}$.

To compute $g_1^{\boldsymbol{c}(\tilde{\mathbb{A}}^*, \boldsymbol{b}(\boldsymbol{w}, 1, \boldsymbol{h}); s, \boldsymbol{s})}$ for $\tilde{\mathbb{A}}^*$, $\mathcal{B}$ computes $g_1^{c_1} = g_1^c$ and $g_1^{c_2} = (g_1^c)^{d+\tilde{\nu}} = (g_1^{dc})(g_1^c)^{\tilde{\nu}}$ using the elements in the instance.

For the ψ row of A^* and its corresponding attribute x_ψ, It randomly selects $\hat{t}_\psi$ and sets $t_\psi = -dcb_\psi + \hat{t}_\psi$. It computes $g_1^{c_{\psi,2}} = (g_1^{dcb_\psi})^{-1} g_1^{\hat{t}_\psi}$. The others can be computed as follows:

- If $\psi \in L$ (i.e., $\rho(\psi) = x_\psi$), $\mathcal{B}$ computes $g_1^{c_{\psi,1}}$ and $g_1^{c_{\psi,2+i'}}; \forall i' \in [N]$ as follows:

$$\begin{aligned} g_1^{c_{\psi,1}} &= g_1^{\delta \lambda_\psi + \zeta t_\psi} \\ &= g_1^{da \sum_{i \in [m]} A^*_{\psi,i} c a^{i-1} + da(d^{-1}) \tilde{\lambda}_\psi + (\tilde{\zeta} + \sum_{(j,k) \in L \times [m]} (A^*_{j,k} a^k / b_j))(-dcb_\psi + \hat{t}_\psi)} \\ &= g_1^{\boxed{\sum_{i \in [m]} A^*_{\psi,i} dca^i} + a\tilde{\lambda}_\psi + \tilde{\zeta}(-dcb_\psi + \hat{t}_\psi)} \\ &\quad \cdot g_1^{-\boxed{\sum_{(j,k) \in L \times [m]} (A^*_{j,k} dca^k b_\psi / b_j)} + \hat{t}_\psi \sum_{(j,k) \in L \times [m]} A^*_{j,k} a^k / b_j} \\ &= g_1^{a\tilde{\lambda}_\psi - \tilde{\zeta} dcb_\psi + \tilde{\zeta} \hat{t}_\psi - \boxed{\sum_{\substack{(j,k) \in L \times [m] \\ j \neq \psi}} (A^*_{j,k} dca^k b_\psi / b_j)} + \hat{t}_\psi \sum_{(j,k) \in L \times [m]} A^*_{j,k} a^k / b_j} \\ &= g_1^{\tilde{\zeta} \hat{t}_\psi} (g_1^a)^{\tilde{\lambda}_\psi} (g_1^{dcb_\psi})^{-\tilde{\zeta}} \prod_{\substack{(j,k) \in L \times [m] \\ j \neq \psi}} (g_1^{dca^k b_\psi / b_j})^{-A^*_{j,k}} \prod_{(j,k) \in L \times [m]} (g_1^{a^k / b_j})^{A^*_{j,k} \hat{t}_\psi}, \end{aligned}$$

$$
\begin{aligned}
g_1^{c_{\psi,2+i'}} &= g_1^{-t_\psi(y_{i'+1}-\rho^*(\psi)^{i'}y_1)} = g_1^{dcb_\psi(y_{i'+1}-\rho^*(\psi)^{i'}y_1)-\hat{t}_\psi(y_{i'+1}-\rho^*(\psi)^{i'}y_1)} \\
&= g_1^{dcb_\psi\left(\tilde{h}_{i'+1}+\boxed{\sum_{(j,k)\in L\times[m]}\rho^*(j)^{i'}A^*_{j,k}\cdot a^k/b_j^2}\right)} \\
&\quad \cdot g_1^{dcb_\psi\left(-\rho^*(\psi)^{i'}\tilde{h}_1-\boxed{\rho^*(\psi)^{i'}\sum_{(j,k)\in L\times[m]}A^*_{j,k}\cdot a^k/b_j^2}\right)} g_1^{\Phi'_1}
\end{aligned}
$$

$$
\begin{aligned}
&= g_1^{dcb_\psi\left(\tilde{h}_{i'+1}-\rho^*(\psi)^{i'}\tilde{h}_1-\boxed{\sum_{\substack{(j,k)\in L\times[m]\\ j\neq\psi}}(\rho^*(\psi)^{i'}-\rho^*(j)^{i'})A^*_{j,k}\cdot a^k/b_j^2}\right)} \cdot g_1^{\Phi'_5} \\
&= (g_1^{dcb_\psi})^{\tilde{h}_{i'+1}-\rho^*(\psi)^{i'}\tilde{h}_1} \prod_{(j,k)\in L\times[m],j\neq\psi} (g_1^{dca^k b_\psi/b_j^2})^{-(\rho^*(\psi)^{i'}-\rho^*(j)^{i'})A^*_{j,k}} \cdot g_1^{\Phi'_1}
\end{aligned}
$$

where $g_1^{\Phi'_1} = g_1^{-\hat{t}_\psi(y_{i'+1}-\rho^*(\psi)^{i'}y_1)}$. $g_1^{\Phi'_1}$ can be computed using $g_1^{y_{i'+1}}$ and $g_1^{y_1}$.

- if $\psi \in L'$, $\mathcal{B}$ computes $g_1^{c_{\psi,1}} = g_1^{\delta\lambda_\psi+\eta y'_1 t_\psi}$ as follows:

$$
\begin{aligned}
g_1^{c_{\psi,1}} &= g_1^{da\sum_{i\in[m]}A^*_{\psi,i}ca^{i-1}+da(d^{-1})\tilde{\lambda}_\psi} \\
&\quad \cdot g_1^{(-dcb_\psi+\hat{t}_\psi)(\tilde{h}'_1\sum_{i\in[n_2]}b_i+\sum_{\substack{(i,k)\in[n_2,m],j\in L'\\ i\neq j}}A^*_{j,k}a^k b_i/b_j^2+\sum_{(j,k)\in L'\times[m]}A^*_{j,k}a^k/b_j)} \\
&= g_1^{\boxed{\sum_{i\in[m]}A^*_{\psi,i}dca^i}+a\tilde{\lambda}_\psi-\tilde{h}'_1\sum_{i\in[n_2]}dcb_\psi b_i} \\
&\quad \cdot g_1^{-\sum_{\substack{(i,k)\in[n_2,m],j\in L'\\ i\neq j}}A^*_{j,k}dca^k b_\psi b_i/b_j^2-\boxed{\sum_{(j,k)\in L'\times[m]}A^*_{j,k}dca^k b_\psi/b_j}} \cdot g_1^{\Phi'_2}
\end{aligned}
$$

$$
\begin{aligned}
&= g_1^{a\tilde{\lambda}_\psi-\tilde{h}'_1\sum_{j\in[n_2]}dcb_\psi b_j} \\
&\quad \cdot g_1^{-\sum_{\substack{(i,k)\in[n_2,m],j\in L'\\ i\neq j}}A^*_{j,k}dca^k b_\psi b_i/b_j^2-\boxed{\sum_{\substack{(j,k)\in L'\times[m]\\ j\neq\psi}}A^*_{j,k}dca^k b_\psi/b_j}} \cdot g_1^{\Phi'_2} \\
&= (g_1^a)^{\tilde{\lambda}_\psi}\prod_{j\in[n_2]}(g_1^{dcb_\psi b_j})^{-\tilde{h}'_1}\prod_{(i,k)\in[n_2,m],j\in L',i\neq j}(g_1^{dca^k b_\psi b_i/b_j^2})^{-A^*_{j,k}} \\
&\quad \cdot \prod_{(j,k)\in L'\times[m],j\neq\psi}(g_1^{dca^k b_\psi/b_j})^{-A^*_{j,k}} \cdot g_1^{\Phi'_2}
\end{aligned}
$$

where $\Phi'_2 = \hat{t}_\psi\eta y_x$. $\mathcal{B}$ can compute $g_1^{\Phi'_2}$ using $g_1^{\eta y_x}$.

Computing $g_1^{c_{\psi,2}}$ is similar to that in the previous case. In this case, $y'_{i'}$ is used instead of $y_{i'}$ and L is replaced by L'. $\mathcal{B}$ computes $g_1^{c_{\psi,2}}$ as follows:

$$
\begin{aligned}
g_1^{c_{\psi,2}} &= g_1^{-t_\psi(y'_{i'+1}-\rho^*(\psi)^{i'}y'_1)} = g_1^{dcb_\psi(y'_{i'+1}-\rho^*(\psi)^{i'}y'_1)-\hat{t}_\psi(y'_{i'+1}-\rho^*(\psi)^{i'}y'_1)} \\
&= (g_1^{dcb_\psi})^{\tilde{h}_{i'+1}-\rho^*(\psi)^{i'}\tilde{h}_1}\prod_{(j,k)\in L\times[m],j\neq\psi}(g_1^{dca^k b_\psi/b_j^2})^{(\rho^*(j)^{i'}-\rho^*(\psi)^{i'})A^*_{j,k}} \cdot g_1^{\Phi'_3}
\end{aligned}
$$

where $g_1^{\Phi'_3} = g_1^{-\hat{t}_\psi(y'_{i'+1}-\rho^*(\psi)^{i'}y'_1)}$. $g_1^{\Phi'_3}$ can be computed using $g_1^{y'_{i'+1}}$ and $g_1^{y'_1}$.

k-type response: When the adversary requests the k-type response to $\mathcal{A}$ for S, $\mathcal{B}$ sets $S' = N(S)$. We let A^*_ψ is the ψth row of the access matrix A^* which is given in the c-type query. By proposition 11 in [13], $\mathcal{B}$ can select a random vector $\boldsymbol{z'} = (1, z'_2, ..., z'_m) \in \mathbb{Z}_p^m$ which satisfies $\langle \boldsymbol{z'}, A_{i'} \rangle = 0$ for all i' such that $\rho(i') \in S'$. Then, $\mathcal{B}$ randomly selects z_1 from $\mathbb{Z}_p$ and sets $\boldsymbol{z} = (z_1, ..., z_m) = z_1 \boldsymbol{z'}$.

$\mathcal{B}$ randomly selects $\hat{r}$ from $\mathbb{Z}_p$ and sets $r_0 = c\hat{r} - \sum_{i\in[m], i\neq 1} z_i a^{n_2+2-i}$ and $r_2 = \sum_{i\in[m]} z_i a^{n_2+1-i}$. r_0 is randomly distributed due to $\hat{r}$ which are not used anywhere else. r_2 is also randomly distributed due to z_1 which does not appear anywhere else.

To compute $g_2^{\boldsymbol{k}(\beta\cdot\alpha, S, \boldsymbol{b}(\boldsymbol{w},1,\boldsymbol{h});\boldsymbol{r})}$ for S. Firstly, $\mathcal{B}$ computes follows:

$$g_2^{d_1} = T^{z_1} \cdot (g_2^{dc})^{\hat{r}} (g_2^{c})^{\tilde{\nu}\hat{r}} \prod_{i\in[m], i\neq 1} (g_2^{a^{n_2+2-i}})^{-\tilde{\nu} z_i}$$

If T is equal to $g_2^{da^{n_2+1}}$, D_1 is equal to $g_2^{\delta r_2 + \nu r_0}$ and $\mathcal{B}$ simulates $\hat{O}_0^{Sel}(n_2)$ because

$$\begin{aligned} g_2^{d_1} &= g_2^{da(\sum_{i\in[m]} z_i a^{n_2+1-i})} g_2^{(d+\tilde{\nu})(c\hat{r} - \sum_{i\in[m], i\neq 1} z_i a^{n_2+2-i})} \\ &= \boxed{\prod_{i\in[m]} g_2^{z_i d a^{n_2+2-i}}} \boxed{\prod_{i\in[m], i\neq 1} (g_2^{z_i d a^{n_2+2-i}})^{-1}} (g_2^{dc})^{\hat{r}} (g_2^{c})^{\tilde{\nu}\hat{r}} \cdot \prod_{i\in[m], i\neq 1} (g_2^{z_i a^{n_2+2-i}})^{-\tilde{\nu}} \\ &= \boxed{(g_2^{da^{n_2+1}})^{z_1}} \cdot (g_2^{dc})^{\hat{r}} (g_2^{c})^{\tilde{\nu}\hat{r}} \prod_{i\in[m], i\neq 1} (g_2^{a^{n_2+2-i}})^{-\tilde{\nu} z_i}. \end{aligned}$$

If T is a random from G_2 and we write it as $g_2^{da^{n_2+1}} g_2^{\tilde{\alpha}}$, the randomness is added to $g_2^{d_1}$ and $\mathcal{B}$ simulates $\hat{O}_1^{Sel}(n_2)$. Then, it computes the others following:

$$g_2^{d_2} = g_2^{r_2} = \prod_{i\in[m]} (g_2^{a^{n_2+1-i}})^{z_i}, \quad g_2^{d_3} = g_2^{r_0} = (g_2^{c})^{\hat{r}} \prod_{i\in[m-1]} (g_2^{a^{n_2+1-i}})^{-z_{i+1}}.$$

• **Compute** $g_2^{d_4}$ *and* $g_2^{d_5}$: $g_2^{d_4} = g_2^{-\zeta r_2 + r_1(a_1 y_1 + ... + a_N y_N)}$ is decomposed to $g_2^{-\zeta r_2}$ and $g_2^{r_1(a_1 y_1 + ... + a_N y_N)}$. First, it derives $g_2^{-\zeta r_2}$ as follows:

$$\begin{aligned} g_2^{-\zeta r_2} &= g_2^{-(\tilde{\zeta} + \sum_{(j,k)\in L\times[m]} A^*_{j,k} \cdot a^k / b_j) \sum_{i\in[m]} z_i a^{n_2+1-i}} \\ &= g_2^{-\tilde{\zeta} \sum_{i\in[m]} z_i a^{n_2+1-i}} g_2^{-\sum_{\substack{(i,j,k)\in[m]\times L\times[m] \\ i\neq k}} A^*_{j,k} z_i \cdot a^{n_2+1+k-i}/b_j} \\ &\quad \cdot g_2^{-\sum_{(i,j)\in[m]\times L} A^*_{j,i} z_i \cdot a^{n_2+1}/b_j} \\ &= g_2^{\Phi'_3} \cdot g_2^{\sum_{j\in L} \langle A^*_j, \boldsymbol{z} \rangle a^{n_2+1}/b_j} = g_2^{\Phi'_3} \cdot \prod_{j\in L, j\notin S} g_2^{\langle A^*_j, \boldsymbol{z} \rangle a^{n_2+1}/b_j} \end{aligned}$$

where $\Phi'_3 = -\tilde{\zeta} \sum_{i\in[m]} z_i a^{n_2+1-i} - \sum_{\substack{(i,j,k)\in[m]\times L\times[m] \\ i\neq k}} A^*_{j,k} z_i \cdot a^{n_2+1+k-i}/b_j$. The last equality of the above equation holds because $\langle A^*_j, \boldsymbol{z} \rangle = 0$ for all j such that $\rho^*(j) \in S$. $g_2^{\phi'_3}$ can be computed since $\{g_2^{a^i}; \forall i \in [n_2]\}$ and $\{g_2^{a^i/b_j}; \forall (i,j) \in [2n_2, n_2], i \neq n_2+1\}$ are given in the instance.

Secondly, to derive $g_2^{r_1(a_1 y_1 + ... + a_N y_N)}$, it creates $\hat{r}_1 \xleftarrow{R} \mathbb{Z}_p$ and sets $r_1 = \hat{r}_1 - \sum_{\substack{(i,i')\in[m]\times L \\ \rho^*(i')\notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{P(\rho^*(i'))}$ where $P(\rho^*(i')) = \Sigma_{y\in S}(\rho^*(i') - y)$. $P(\rho^*(i')) \neq 0$ due to $\rho^*(i') \notin S$. $\mathcal{B}$ computes

$$
\begin{aligned}
&g_2^{r_1(a_1y_1+...+a_Ny_N)}\\
&= g_2^{\hat{r}_1(a_1y_1+...+a_Ny_N)-\sum_{\substack{(i,i')\in[m]\times L\\ \rho^*(i')\notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{P(\rho^*(i'))}(a_1\tilde{y}'_1+...+a_N\tilde{y}'_N)}\\
&\quad \cdot g_2^{-\sum_{\substack{(i,i')\in[m]\times L\\ \rho^*(i')\notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{P(\rho^*(i'))} \sum_{(j,k)\in L\times[m]} P(\rho^*(j)) A^*_{j,k} a^k/b_j^2)}\\
&= g_2^{\Phi'_4} \cdot g_2^{-\sum_{\substack{(i,i',j,k)\in[m]\times L^2\times[m]\\ \rho^*(i')\notin S}} \frac{P(\rho^*(j))}{P(\rho^*(i'))} A^*_{j,k} z_i a^{n_2+1+k-i} b_{i'}/b_j^2}\\
&= g_2^{\Phi'_4} \cdot g_2^{-\sum_{\substack{(i,i',j,k)\in[m]\times L^2\times[m]\\ \rho^*(i')\notin S,(j\neq i')\vee(i\neq k)}} \frac{P(\rho^*(j))}{P(\rho^*(i'))} A^*_{j,k} z_i a^{n_2+1+k-i} b_{i'}/b_j^2}\\
&\quad \cdot g_2^{-\sum_{\substack{(i,i',j,k)\in[m]\times L^2\times[m]\\ \rho^*(i')\notin S,(j=i')\wedge(i=k)}} \frac{P(\rho^*(j))}{P(\rho^*(i'))} A^*_{j,k} z_i a^{n_2+1+k-i} b_{i'}/b_j^2}\\
&= g_2^{\Phi'_4} \cdot g_2^{\Phi'_5} \cdot g_2^{-\sum_{j\in L,\, \rho^*(j)\notin S} \langle A^*_j, \boldsymbol{z}\rangle a^{n_2+1}/b_j}
\end{aligned}
$$

where $\Phi'_4 = \hat{r}_1(a_1y_1+...+a_Ny_N)-\sum_{\substack{(i,i')\in[m]\times L\\ \rho^*(i')\notin S}} \frac{z_i b_{i'} a^{n_2+1-i}}{P(\rho^*(i'))}(a_1\tilde{y}'_1+...+a_N\tilde{y}'_N)$ and $\Phi'_5 = -\sum_{\substack{(i,i',j,k)\in[m]\times L^2\times[m]\\ \rho^*(i')\in S,(j\neq i')\vee(i\neq k)}} \frac{P(\rho^*(j))}{P(\rho^*(i'))} A^*_{j,k} z_i$ Both $g_2^{\Phi'_4}$ and $g_2^{\Phi'_5}$ can be computed since $g_2^{y_1}$, ..., $g_2^{y_w}$ can be computed and $\{g_2^{a^i b_j}; \forall(i,j)\in[n_2,n_2]\}$ and $\{g_2^{a^i b_j/b_{j'}^2}; \forall(i,j,j')\in[2n_2,n_2,n_2], j\neq j'\}$ are given in the instance.

Because $g_2^{-\sum_{j\in L,\, \rho^*(j)\notin S} \langle A^*_j, \boldsymbol{z}\rangle a^{n_2+1}/b_j}$ is cancelled out when $\mathcal{B}$ computes $g_2^{d_4} = g_2^{-\zeta r_2} g_2^{r_1(a_1y_1+...+a_Ny_N)}$, it can compute $g_2^{d_4}$ and $g_2^{d_5}$ as follows:

$$
g_2^{d_4} = g_2^{\Phi'_3} g_2^{\Phi'_4} g_2^{\Phi'_5}, \quad g_2^{d_5} = g_2^{\hat{r}_\psi} \prod_{\substack{(i,i')\in[m]\times L\\ \rho^*(i')\notin S}} (g_2^{b_{i'} a^{n_2+1-i}})^{\frac{z_i}{P(\rho^*(i'))}}.
$$

• **Compute** $g_2^{d'_4}$ *and* $g_2^{d'_5}$: In order to compute $g_2^{d'_4} = (g_2^{\eta(a_1y'_1+...+a_Ny'_N)})^{r_2}$ and $g_2^{d'_5} = (g_2^{\eta})^{r_2}$, $\mathcal{B}$ first sets $g_2^{d'_4} \leftarrow 1_{G_2}$ and $g_2^{d'_5} \leftarrow 1_{G_2}$. Then, it updates $g_2^{d'_4}$ and $g_2^{d'_5}$ by multiplying $g_2^{b_\pi r(a_1y'_1+...+a_Ny'_N)}$ and $g_2^{b_\pi r}$ for all $\pi\in[n_2]$. This process allows $\mathcal{B}$ to compute $\prod_{\pi\in[n_2]} g_2^{b_\pi r(a_1y'_1+...+a_Ny'_N)} = g_2^{\eta r(a_1y'_1+...+a_Ny'_N)}$ and $\prod_{\pi\in[n_2]} g_2^{b_\pi r} = g_2^{\eta r}$.

It is achieved by repeating the following **Update** process for all $\pi\in[n_2]$.

Update: First, $\mathcal{B}$ can compute $g_2^{b_\pi r}$ because $g_2^{b_\pi r} = \prod_{i\in[m]}(g_2^{a^{n_2+1-i} b_\pi})^{-z_i}$ and $g_2^{a^i b_j}$ is given in the instance.

There are three cases to compute $g_2^{b_\pi r(w_\psi y_x + y_y)}$ as follows:

[Case 1] If $\pi\in L' \wedge \rho(\pi) = x'_\psi \wedge x_\psi = w_\psi \in S$,

$$
\begin{aligned}
&g_2^{b_\pi r(a_1y'_1+...+a_Ny'_N)}\\
&= g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)(\sum_{i\in[N]} a_i \tilde{y}'_i)}\\
&\quad \cdot g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)(\boxed{\sum_{(i,j,k)\in[N]\times L'\times[m]} a_i \rho(j)^i a^k A^*_{j,k}/b_j^2})}\\
&= g_2^{\Phi'_6} \cdot g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)\boxed{\sum_{\substack{(i,j,k)\in[N]\times L'\times[m]\\ j\neq\pi}} a_i \rho(j)^i A^*_{j,k} a^k/b_j^2}}\\
&= g_2^{\Phi'_6} \prod_{(i,i',j,k)\in[m]\times[N]\times L'\times[m], j\neq\pi} (g_2^{a^{n_2+1+k-i} b_\pi/b_j^2})^{a_{i'} \rho(j)^{i'} A^*_{j,k} z_i}
\end{aligned}
$$

where $\varPhi'_6 = (\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)\cdot(\sum_{i\in[N]} a_i \tilde{y}'_i)$. $\mathcal{B}$ can compute $g_2^{\varPhi'_6}$ since $\{g_2^{a^i b_j}; \forall(i,j)\in[n_2,n_2]\}$ is given in the instance. a^k/b_π^2 was cancelled out in the boxed terms since $\sum_{(i,k)\in[N]\times[m]} a_i\rho(\pi)^i A^*_{\pi,k} a^k/b_\pi^2 = 0$. In the above equation, $g_2^{a^{n_2+1}/b_\pi}$ which is not given in the instance does not appear since $j\neq\pi$. Therefore, $g_2^{d'_{\pi,1}}$ can be computed.

[Case 2] If $\pi\in L' \wedge \rho(\pi) = x'_\psi \wedge x_\psi \notin S$,

$$\begin{aligned}
&g_2^{b_\pi r(a_1y'_1+\ldots+a_Ny'_N)}\\
&= g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)(\sum_{i\in[N]} a_i\tilde{y}'_i)}\\
&\quad\cdot g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)(\boxed{\sum_{(i,j,k)\in[N]\times L'\times[m]} a_i\rho(j)^i a^k A^*_{j,k}/b_j^2})}\\
&= g_2^{\varPhi'_6}\cdot g_2^{\sum_{\substack{(i,i',j,k)\in[m]\times[N]\times L'\times[m]\\(k\neq i)\vee(j\neq\pi)}} a_{i'}\rho(j)^{i'} A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi/b_j^2}\\
&\quad\cdot g_2^{\sum_{\substack{(i,i',j,k)\in[m]\times[N]\times L'\times[m]\\(k=i)\wedge(j=\pi)}} a_{i'}\rho(j)^{i'} A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi/b_j^2}\\
&= g_2^{\varPhi'_6}\cdot g_2^{\varPhi'_7}\cdot g_2^{\sum_{i,i'\in[m]\times([N])} a_{i'}\rho(\pi)^{i'} A^*_{\pi,i} z_i a^{n_2+1}/b_\pi} \qquad (17)\\
&= g_2^{\varPhi'_6}\cdot g_2^{\varPhi'_7}
\end{aligned}$$

where $\varPhi'_7 = \sum_{\substack{(i,i',j,k)\in[m]\times[N]\times L'\times[m]\\(k\neq i)\vee(j\neq\pi)}} a_{i'}\rho(j)^{i'} A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi/b_j^2$. The equality of (17) holds since $\langle A^*_\pi, \boldsymbol{z}\rangle = 0$ for all π such that $\rho^*(\pi)\in S'$ (i.e. $\rho^*(\pi)\notin S$). $g_2^{\varPhi'_7}$ can be computed since $\{g_2^{a^i b_j/b_{j'}^2}; \forall(i,j,j')\in[2n_2,n_2,n_2], j\neq j'\}$ are given.

[Case 3] If $\pi\notin L'$,

$$\begin{aligned}
&g_2^{b_\pi r(a_1y'_1+\ldots+a_Ny'_N)}\\
&= g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)(\sum_{i\in[N]} a_i\tilde{y}'_i)}\\
&\quad\cdot g_2^{(\sum_{i\in[m]} z_i a^{n_2+1-i} b_\pi)(\boxed{\sum_{(i,j,k)\in[N]\times L'\times[m]} a_i\rho(j)^i a^k A^*_{j,k}/b_j^2})}\\
&= g_2^{\varPhi'_6}\cdot g_2^{\sum_{(i,i',j,k)\in[m]\times[N]\times L'\times[m]} a_{i'}\rho(j)^{i'} A^*_{j,k} z_i a^{n_2+1+k-i} b_\pi/b_j^2}\\
&= g_2^{\varPhi'_6} \prod_{(i,i',j,k)\in[m]\times[N]\times L'\times[m]} (g_2^{a^{n_2+1+k-i} b_\pi/b_j^2})^{a_{i'}\rho(j)^{i'} A^*_{j,k} z_i}.
\end{aligned}$$

It should be noted that $g_2^{a^{n_2+1+k-i}/b_\pi}$ which was not given in the instance does not appear in the above equation because $j\in L'$.

Now, it updates $g_2^{d'_{\psi,1}} \leftarrow g_2^{d'_{\psi,1}}\cdot(g_2^{k'_{\pi,1}})^{1/|S|}$ and $g_2^{d'_{\psi,2}} \leftarrow g_2^{d'_{\psi,2}}\cdot(g_2^{k'_{\pi,2}})^{1/|S|}$.

□

B Generic Security of Assumptions

We will prove our assumptions are secure in the generic group model. Our assumptions are based on n-(A), n-(B) assumptions of [31]. However, in our assumptions, the indistinguishable element is in G_2. Hence, proving the generic security of our assumptions is more complicated than proving n-(A), n-(B) assumptions which have the indistinguishable element in G_T.

B.1 Converting Assumptions

We prove the security of assumptions in a generic way instead of providing separate proofs for each assumption. Our proof is inspired by [22], but we use the assumptions of selective security for the generalized proof. First, we formalize both assumptions. We then show that an assumption of which the indistinguishable element is in G_2 can be created and proved from an assumption which is used to prove selective security such as n-(A), n-(B) assumptions.

We describe our notation. We set $\boldsymbol{h} = (h_1, ..., h_\ell) \in \mathbb{Z}_p^\ell$ and let $\boldsymbol{h}_{i|x}$ denote $(h_1, ..., h_{i-1}, x, h_{i+1}, ..., h_\ell)$ in which the ith coordinate of $\boldsymbol{h}$ is replaced by x and the other coordinates are unchanged. We define $\boldsymbol{z}(\boldsymbol{h})$ to be the set of ratio monomials which are outputs of rational functions of $\boldsymbol{h}$. We also define $v(\boldsymbol{h})$ to be a rational function outputting a monomial and there exists $i \in [\ell]$ such that $v(\boldsymbol{h}_{i|x}) = x \cdot v(\boldsymbol{h}_{i|1})$. Using $\boldsymbol{z}$ and v, we define the following two assumptions:

Assumption$_{Sel}(\boldsymbol{z}, v, \ell)$ If a group generator $\mathcal{G}$, we define following distribution

$$\mathbb{G} = (p, G, G_T, e) \xleftarrow{R} \mathcal{G}, \quad g \xleftarrow{R} G, \quad \boldsymbol{h} \xleftarrow{R} \mathbb{Z}_p^\ell$$

$$Z = \{g\} \cup \{g^z | z \in \boldsymbol{z}(\boldsymbol{h})\}, T_0 = e(g,g)^{v(\boldsymbol{h})}, T_1 \xleftarrow{R} G_T$$

We define the advantage of $\mathcal{A}$ in breaking this assumption to be:

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

We say that $\mathcal{G}$ satisfies an algorithm $\mathcal{A}$ breaks Assumption$_{Sel}(\boldsymbol{z}, v)$ if $Adv_{\mathcal{G},\mathcal{A}}^{Sel}(\lambda)$ is a negligible function of λ for any PPT.

Assumption$_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v, \ell)$ If a group generator $\mathcal{G}$, we define following distribution

$$\mathbb{G} = (p, G_1, G_2, G_T, e) \xleftarrow{R} \mathcal{G}, f_1 \xleftarrow{R} G_1, f_2 \xleftarrow{R} G_2, c, d, h_1, ..., h_{i-1}, h_{i+1}, ..., h_\ell \xleftarrow{R} \mathbb{Z}_p$$

$$Z_1 = \{f_1, f_1^c\} \cup \{f_1^{z_1} | z_1 \in \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\} \text{ and } Z_2 = \{f_2, f_2^c\} \cup \{f_2^{z_2} | z_2 \in \boldsymbol{z}_2(\boldsymbol{h}_{i|dc})\}$$

$$T_0 = f_2^{v(\boldsymbol{h}_{i|d})}, T_1 \xleftarrow{R} G_2$$

where e is an asymmetric pairing such that $e : G_1 \times G_2 \to G_T$.

We define the advantage of $\mathcal{A}$ in breaking this assumption to be:

$$Adv_{\mathcal{G},\mathcal{A}}^{SF,Asym}(\lambda) := |Pr[A(Z, T_0 = 1] - Pr[A(D, T_1) = 1]|.$$

We say that $\mathcal{G}$ satisfies an algorithm $\mathcal{A}$ breaks Assumption$_{SF}(\boldsymbol{z}, v)$ if $Adv_{\mathcal{G},\mathcal{A}}^{SF}(\lambda)$ is a negligible function of λ for any PPT.

For $\boldsymbol{h} \in \mathbb{Z}_p^\ell$, we let $\mathcal{Z}_1$ and $\mathcal{Z}_2$ denote $\{1, c\} \cup \{z_1 | z_1 \in \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\}$ and $\{1, c\} \cup \{z_2 | z_2 \in \boldsymbol{z}_2(\boldsymbol{h}_{i|dc})\}$, respectively. Then, we define $E(\mathcal{Z}_1, \mathcal{Z}_2)$ be $\{xy | (x, y) \in \mathcal{Z}_1 \times \mathcal{Z}_2\}$, the set of all possible pairwise products between $\mathcal{Z}_1$ and $\mathcal{Z}_2$. Therefore, $E(\mathcal{Z}_1, \mathcal{Z}_2)$ represents all exponents of elements in an asymmetric pairing can be obtained by $\{f_1, f_2, f_1^c, f_2^c\} \cup \{f_1^{z_1}, f_2^{z_2} | z_1 \in \boldsymbol{z}_1(\boldsymbol{h}_{i|dc}), z_2 \in \boldsymbol{z}_2(\boldsymbol{h}_{i|dc})\}$.

Proposition 1. *If Assumption$_{sel}(\boldsymbol{z}, v, \ell)$ holds, Assumption$_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v, \ell)$ also holds for all rational functions $\boldsymbol{z}_1$ and $\boldsymbol{z}_2$ such that $\boldsymbol{z}_1(\boldsymbol{h}), \boldsymbol{z}_2(\boldsymbol{h}) \subseteq \boldsymbol{z}(\boldsymbol{h})$ for all $\boldsymbol{h} \in \mathbb{Z}_p^\ell$.*

Proof: To prove the proposition, we first prove the following claim:

Claim: For each function $M \in \mathcal{Z}_1$, the product $M \cdot v(\boldsymbol{h}_{i|d})$ is not in $E(\mathcal{Z}_1, \mathcal{Z}_2) \cup v(\boldsymbol{h}_{i|d}) \cdot (\mathcal{Z}_1 \setminus \{M\})$ where $v(\boldsymbol{h}_{i|d}) \cdot (\mathcal{Z}_1 \setminus \{M\})$ is the set formed by multipying $v(\boldsymbol{h}_{i|d})$ to all elements in $\mathcal{Z}_1 \setminus \{M\}$.

It is obvious that $M \cdot v(\boldsymbol{h}_{i|d})$ cannot be in $v(\boldsymbol{h}_{i|d})(\mathcal{Z}_1 \setminus \{M\})$. Therefore, we will show that $M \cdot v(\boldsymbol{h}_{i|d})$ for all $M \in \mathcal{Z}_1$ is also not in $E(\mathcal{Z}_1, \mathcal{Z}_2)$. First, we can compute $E(\mathcal{Z}_1, \mathcal{Z}_2)$ as follows:

$$\{1, c, c^2\} \cup c \cdot \boldsymbol{z}_1(\boldsymbol{h}_{i|dc}) \cup c \cdot \boldsymbol{z}_2(\boldsymbol{h}_{i|dc}) \cup \{z_1 \cdot z_2 | (z_1, z_2) \in \boldsymbol{z}_1(\boldsymbol{h}_{i|dc}) \times \boldsymbol{z}_2(\boldsymbol{h}_{i|dc})\}$$

Since all elements of $\mathcal{Z}_1$, $\mathcal{Z}_2$ and $v(\boldsymbol{h}_{i|d})$ are a ratio of monomials, all elements of $E(\mathcal{Z}_1, \mathcal{Z}_2) \cup v(\boldsymbol{h}_{i|d}) \cdot (\mathcal{Z}_1 \setminus \{M\})$ are also a ratio of monomials. Therefore, $M \cdot v(\boldsymbol{h}_{i|d})$ is also a monomial and we will show that it is not in $E(\mathcal{Z}_1, \mathcal{Z}_2) \cup v(\boldsymbol{h}_{i|d})(\mathcal{Z}_1 \setminus \{M\})$. Because $M \in \mathcal{Z}_1$, $M \cdot v(\boldsymbol{h}_{i|d})$ is in $(v(\boldsymbol{h}_{i|d}) \cdot \mathcal{Z}_1)$. Then, we compute $v(\boldsymbol{h}_{i|d}) \cdot (\mathcal{Z}_1)$:

$$v(\boldsymbol{h}_{i|d}) \cdot \mathcal{Z}_1 := \{v(\boldsymbol{h}_{i|d}), c \cdot v(\boldsymbol{h}_{i|d}), v(\boldsymbol{h}_{i|d}) \cdot z_1 | z_1 \in \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\}.$$

Case 1: $M \cdot v(\boldsymbol{h}_{i|d})$ is in $\{v(\boldsymbol{h}_{i|d}), v(\boldsymbol{h}_{i|d}) \cdot z_1 | z_1 \in \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\}$.

For all monomials in $E(\mathcal{Z}_1, \mathcal{Z}_2)$, the degree of d is less than the degree of c because d is always accompanied by c and c^{-1} is not in $\mathcal{Z}_1$ and $\mathcal{Z}_2$. It means that, if $M \cdot v(\boldsymbol{h}_{i|d})$ is in $\{v(\boldsymbol{h}_{i|d}), v(\boldsymbol{h}_{i|d}) \cdot \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\}$, it cannot have a common element with $E(\mathcal{Z}_1, \mathcal{Z}_2)$ because $M \cdot v(\boldsymbol{h}_{i|d})$ is a polynomial of which the degree of d is larger than the degree of c.

Case 2: $M \cdot v(\boldsymbol{h}_{i|d}) = c \cdot v(\boldsymbol{h}_{i|d})$ (i.e. M is equal to c).

By the definition of function v, $c \cdot v(\boldsymbol{h}_{i|d})$ is equal to $dc \cdot v(\boldsymbol{h}_{i|1})$, and $v(\boldsymbol{h}_{i|1})$ dose not contain any of d and c. Hence, it is not in $\{1, c, c^2\}$ since they do not include d, obviously. $dc \cdot v(\boldsymbol{h}_{i|1})$ is not in $c \cdot \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})$ or $c \cdot \boldsymbol{z}_2(\boldsymbol{h}_{i|dc})$ since the degree of c is larger than the degree of d for all elements of $c \cdot \boldsymbol{z}_1(\boldsymbol{h}_{i|dc})$ and $c \cdot \boldsymbol{z}_2(\boldsymbol{h}_{i|dc})$. Moreover, $dc \cdot v(\boldsymbol{h}_{i|1})$ which is equal to $v(\boldsymbol{h}_{i|dc})$ is not in $\{\boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\} \cdot \{\boldsymbol{z}_2(\boldsymbol{h}_{i|dc})\}$ since the Assumption$_{Sel}(\boldsymbol{z}, v)$ holds. In detail, Assumption$_{Sel}(\boldsymbol{z}, v)$ implies that $dc \cdot v(\boldsymbol{h}_{i|1}) = v(\boldsymbol{h}_{i|dc})$ is not in $\{1, \boldsymbol{z}(\boldsymbol{h}_{i|dc})\} \cdot \{1, \boldsymbol{z}(\boldsymbol{h}_{i|dc})\}$ which is equal to $\{1, \boldsymbol{z}(\boldsymbol{h}_{i|dc})\} \cup (\{\boldsymbol{z}(\boldsymbol{h}_{i|dc})\} \cdot \{\boldsymbol{z}(\boldsymbol{h}_{i|dc})\})$. Since $\boldsymbol{z}_1$, $\boldsymbol{z}_2 \subseteq \boldsymbol{z}$, $\{\boldsymbol{z}_1(\boldsymbol{h}_{i|dc})\} \cdot \{\boldsymbol{z}_2(\boldsymbol{h}_{i|dc})\}$ is also a subset of $\{\boldsymbol{z}(\boldsymbol{h}_{i|dc})\} \cdot \{\boldsymbol{z}(\boldsymbol{h}_{i|dc})\}$.

By the above claim, if we take any element from Z_1 of Assumption$_{SF}^{Asym}$ $(\boldsymbol{z}_1, \boldsymbol{z}_1, v)$ and compute pairing with $T_\beta \in G_2$, there do not exist any pairing computations possible to compare the result with to distinguish whether β is 0 or 1 in the generic group models. $\{e(a,b); \forall a, b \in Z_1 \times Z_2\} \cap \{e(a,T); \forall a \in Z_1\} = \emptyset$

Therefore, Assumption$_{SF}^{Asym}$ $(\boldsymbol{z}_1, \boldsymbol{z}_1, v)$ holds. □

B.2 The Generic Security of Assumptions A1-(n) and A2-(n)

Using Proposition 1, we can prove the security of our assumptions in the generic group model. In detail, our assumptions can be proved using n-(A) and n-(B) assumptions of [31]. If we set n-(A) and n-(B) assumptions as Assumption$_{Sel}(\boldsymbol{z}, v, \ell)$ and ours as Assumption$_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v, \ell)$. The security is proved directly by Proposition 1.

Lemma A. A1-(n) is secure in the generic group model.
Proof: To prove this assumption, we first define a new assumption which can be used as Assumption$_{sel}(\boldsymbol{z}, v, \ell)$ of Proposition 1. such that $Z_1, Z_2 \in \boldsymbol{z}(\boldsymbol{h}_{1|dc})$ and $v(\boldsymbol{h}_{1|d}) = dyz$ where $\boldsymbol{h} = (s, x, y, z, a_1, ..., a_n, b_1, ..., b_n) \in \mathbb{Z}_p^{2n+4}$ (i.e. $\ell = 2n+4$) and Z_1 and Z_2 are the sets appearing in A1-(n). By setting $\boldsymbol{z}(\boldsymbol{h}_{1|dc}) = Z_1 \cup Z_2$, we can define Assumption$_{sel}(\boldsymbol{z}, v, \ell)$ as follows:

Assumption$_{sel}(\boldsymbol{z}, v, \ell)$. If a group generator $\mathcal{G}$ and a positive integer n is given, we define following distribution

$$\mathbb{G} = (p, G, G_T, e) \xleftarrow{R} \mathcal{G}, \quad g \xleftarrow{R} G, \quad \boldsymbol{h} = (d, c, x, y, z, a_1, ..., a_n, b_1, ..., b_n) \xleftarrow{R} \mathbb{Z}_p^\ell$$

$$D := \{g\} \cup \{g^z | z \in \boldsymbol{z}(h)\}$$

$$\begin{aligned} \text{where } \boldsymbol{z(h)} = \{ & s, x, y, z, (sz)^2, a_i, sza_i, sz/a_i, (s)^2 za_i, y/a_i^2, y^2/a_i^2 \ \forall i \in [n], \\ & a_i sza_i/a_j, ya_i/a_j^2, syza_i/a_j^2, (sz)^2 a_i/a_j \ \forall (i,j) \in [n,n], i \neq j, \\ & b_i, xb_i, xzb_i, zb_i, b_ib_j, sy/b_i^2, sxy/b_i^2, sxyb_i/b_j^2 \ \forall (i,j) \in [n,n], \\ & zb_ib_j, syb_i/b_j^2, syb_ib_j/b_k^2, syb_i^2/b_j^2 \ \forall (i,j,k) \in [n,n,n], i \neq j \} \end{aligned}$$

$$T_0 = e(g,g)^{syz} \text{ and } T_1 \xleftarrow{R} G_T$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption$_{sel}(\boldsymbol{z}, v, \ell)$ to be

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

Claim: Assumption$_{sel}(\boldsymbol{z}, v, \ell)$ is secure in the generic group model.

Proof of the claim: This claim is trivially holds by n-(A) Assumption of [31]. Compared to n-(A) Assumption, our assumption has only additional elements where x is appears in D and there is no element x^{-1}. This means that all possible pairing computations with newly added elements in our assumption must have x in their exponents. However, x does not appear T_0. Hence, if n-(A) Assumption is secure in the generic group model, n-(A) Assumption is also secure in the generic group model. □

If we define $\text{Assumption}_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v)$ of Proposition 1 by setting $\boldsymbol{z}_1(\boldsymbol{h}_{1|dc}) = Z_1, \boldsymbol{z}_2(\boldsymbol{h}_{1|dc}) = Z_2, v(\boldsymbol{h}_{1|d}) = dyz$ and $\ell = 2n + 4$. This is possible because $\boldsymbol{z}_1(\boldsymbol{h}), \boldsymbol{z}_2(\boldsymbol{h}) \in \boldsymbol{z}(\boldsymbol{h})$ and $\boldsymbol{z}(\boldsymbol{h}_{1|dc}) = Z_1 \cup Z_2$. Then, $\text{Assumption}_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v)$ is identical with A1-(n). Therefore, A1-(n) is secure in the generic group model by Proposition 1. □

Lemma B. A2-(n) is secure in the generic group model.
Proof: We use the n-(B) Assumption of [31] to prove the security of A2-(n). From the n-(B) Assumption, we can set $\ell = n + 2$, $v(\boldsymbol{h}) = sa^{n+1}$ and

$$\begin{aligned}
\boldsymbol{z}(\boldsymbol{h}) := \{ & s, a^i, b_j, sb_j, sb_ib_j, a^ib_j, a^i/b_j^2 \quad \forall(i,j) \in [n,n] \\
& a^i/b_j \qquad \forall(i,j) \in [2n,n], i \neq n+1 \\
& a^ib_j/b_{j'}^2 \qquad \forall(i,j,j') \in [2n,n,n], j \neq j' \\
& sa^ib_j/b_{j'}, sa^ib_j/b_{j'}^2 \qquad \forall(i,j,j') \in [n,n,n], j \neq j' \\
& sa^ib_jb_j'/b_{j''}^2 \qquad \forall(i,j,j',j'') \in [n,n,n,n], j \neq j', j' \neq j'' \},
\end{aligned}$$

where $\boldsymbol{h} = (s, a, b_1, ..., b_n)$. Since $v(\boldsymbol{h}_{1|d}) = da^{n+1} = d \cdot v(\boldsymbol{h}_{1|1})$ where $\boldsymbol{h}_{1|d} := (d, a, b_1, ..., b_n)$, we can set the n-(B) Assumption as $\text{Assumption}_{sel}(\boldsymbol{z},\ v,\ \ell)$.

To show the our A2-(n) is a corresponding conversion of $\text{Assumption}_{sel}(\boldsymbol{z},\ v,\ \ell)$ which can be denoted as $\text{Assumption}_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v)$. First, we set $\boldsymbol{h}_{1|dc} := (dc, a, b_1, ..., b_n)$. Then, we define $\boldsymbol{z}_1(\boldsymbol{h}_{1|dc}) := Z_1$ and $\boldsymbol{z}_2(\boldsymbol{h}_{1|dc}) := Z_2$ where Z_1 and Z_2 are the sets appearing in A2-(n). Since both $\boldsymbol{z}_1(\boldsymbol{h}_{1|dc})$ and $\boldsymbol{z}_2(\boldsymbol{h}_{1|dc})$ are subsets of $\boldsymbol{z}(\boldsymbol{h}_{1|dc})$, our A2-(n) is identical to $\text{Assumption}_{SF}^{Asym}(\boldsymbol{z}_1, \boldsymbol{z}_2, v)$. It means that A2-(n) is also secure in the generic group model by Proposition 1 because the n-(B) Assumption is secure in the generic group model. □

Lemma C. (Asymmetric) n-DBDHE is secure in the generic group model.
Proof: Because n-DBDHE assumption [10] is secure in symmetric pairing, our (Asymmetric) n-DBDHE is also secure by Proposition 1. □