Forward-Secure Hierarchical Predicate Encryption*

Juan Manuel González Nieto[1] and Mark Manulis[2] and Dongdong Sun[1]

[1]Queensland University of Technology, Brisbane QLD 4001, Australia
j.gonzaleznieto@qut.edu.au, dd.sun@student.qut.edu.au
[2]University of Surrey, Guildford, United Kingdom
mark@manulis.eu

Abstract. Secrecy of decryption keys is an important pre-requisite for security of any encryption scheme and compromised private keys must be immediately replaced. *Forward Security (FS)*, introduced to Public Key Encryption (PKE) by Canetti, Halevi, and Katz (Eurocrypt 2003), reduces damage from compromised keys by guaranteeing confidentiality of messages that were encrypted prior to the compromise event. The FS property was also shown to be achievable in (Hierarchical) Identity-Based Encryption (HIBE) by Yao, Fazio, Dodis, and Lysyanskaya (ACM CCS 2004). Yet, for emerging encryption techniques, offering flexible access control to encrypted data, by means of functional relationships between ciphertexts and decryption keys, FS protection was not known to exist.

In this paper we introduce FS to the powerful setting of *Hierarchical Predicate Encryption (HPE)*, proposed by Okamoto and Takashima (Asiacrypt 2009). Anticipated applications of FS-HPE schemes can be found in searchable encryption and in fully private communication. Considering the dependencies amongst the concepts, our FS-HPE scheme implies forward-secure flavors of Predicate Encryption and (Hierarchical) Attribute-Based Encryption.

Our FS-HPE scheme guarantees forward security for plaintexts and for attributes that are hidden in HPE ciphertexts. It further allows delegation of decrypting abilities at any point in time, independent of FS time evolution. It realizes zero-inner-product predicates and is proven adaptively secure under standard assumptions. As the "cross-product" approach taken in FS-HIBE is not directly applicable to the HPE setting, our construction resorts to techniques that are specific to existing HPE schemes and extends them with what can be seen as a reminiscent of binary tree encryption from FS-PKE.

1 Introduction

PREDICATE ENCRYPTION. We focus on the notion of Predicate Encryption (PE), formalized by Katz, Sahai, and Waters [24], building on Hidden Vector Encryption (HVE) [9], and further studied in [25, 27, 28, 30, 31, 36, 37]. In PE schemes users' decryption keys are associated with *predicates* f and ciphertexts encode *attributes* a that are specified during the encryption procedure. A user can successfully decrypt if and only if $f(a) = 1$. Otherwise, the decryption process preserves *plaintext hiding* and thus leaks no information about the encrypted message. Unlike Attribute-Based Encryption (ABE) [4, 14, 18, 32] that imposes the same requirement, PE schemes have a distinguished privacy goal of *attribute hiding* to prevent ciphertext leaking attributes. Existing PE schemes typically realize concrete predicates f. For example, predicates based on the inner product of vectors (over a field or ring) — Inner-Product Encryption (IPE) [24] — are particularly powerful since they can be used to evaluate a large class of predicates, including conjunctions or disjunctions of equality tests, comparisons, and subset tests, or more generally, arbitrary CNF or DNF formulae. In IPE schemes, attributes are represented by a vector $\overrightarrow{y}$ while the choice of another vector $\overrightarrow{x}$ defines the predicate $f_{\overrightarrow{x}}$ such that $f_{\overrightarrow{x}}(\overrightarrow{y}) = 1$ iff the inner-vector product $\overrightarrow{x} \cdot \overrightarrow{y} = 0$. While the original scheme from [24] was proven to be selectively secure under non-standard assumptions, recent result of Lewko *et al.* [25] provided more sophisticated PE constructions achieving (stronger) adaptive security under non-standard assumptions. Furthermore, Okamoto and Takashima [28] investigated Functional Encryption that is adaptive security under standard assumptions. In [25, 27] the authors also explored constructions of Hierarchical PE (HPE) schemes providing their users with the ability to delegate their decryption keys down the hierarchy by restricting predicates associated to the delegated keys and by this restricting the abilities of lower-level users to decrypt. It should be noted that existing PE (and ABE) schemes emerged from Identity-Based Encryption (IBE) [8, 35] and the majority of these schemes are pairing-based.

* This is full version of the paper that appears in Proceedings of the 5th International Conference on Pairing-Based Cryptography (Pairing 2012) published as post-proceedings by Springer.

FORWARD SECURITY. Forward Security (FS) offers meaningful protection in cryptographic applications with long-term (aka. static) private keys in the unfortunate case when these keys become compromised. Being a standard requirement in authenticated key exchange protocols, where it also takes its origin [15, 19], forward security has further been explored in digital signatures [3, 21] and in public key encryption (PKE) [11]; see [21] for a nice survey and strong motivation of forward security. The concept of *time evolution* is central to forward security since from the moment the private key is exposed the intended security goals can no longer be guaranteed and the key must be changed. FS aims to tame potential damage by offering protection with respect to *earlier* time periods. For example, in forward secure digital signatures signing keys that are exposed in one time period cannot be used to forge signatures related to prior time periods. Similarly, in the case of forward secure encryption decryption keys used in one time period cannot be used to decrypt ciphertexts generated in the past.

The first forward-secure PKE scheme, due to Canetti, Halevi, and Katz [11], was built from the technical tool, called *binary tree encryption* [23], which in turn is implied by Hierarchical IBE (HIBE) [17, 20] by considering identities as nodes of the tree and restricting the intermediate nodes to have exactly two descendants: a parent node with identity string $\mathtt{id} \in \{0,1\}^{\ell}$ is split into two child nodes with identities $\mathtt{id}0$, $\mathtt{id}1 \in \{0,1\}^{\ell+1}$. For each node $\mathtt{id}$ there exists a secret key $SK_{\mathtt{id}}$, which can be used to derive secret keys $SK_{\mathtt{id}0}$ and $SK_{\mathtt{id}1}$ in a one-way fashion. The intuition behind FS-PKE is to split the entire lifetime of the scheme into N time periods and construct a binary tree with depth $\log N$, where each node corresponds to a unique time period. In order to encrypt a message for some time period $i \in [1, N]$ one uses the master public key of HIBE and the identity string $\mathtt{id}_i$ of the node i. At any period $i \in [1, N]$ the private decryption key of the user contains the secret key $SK_{\mathtt{id}_i}$ as well as secret keys for all right siblings of the nodes on the path from the root to node i. The latter keys can be used to derive secret keys $SK_{\mathtt{id}_j}$ for all subsequent periods $j \in [i, N]$. The actual FS property is obtained by erasing $SK_{\mathtt{id}_i}$ (and all secret keys that can be used to derive it) from the private key upon transition to period $i + 1$.

These ideas were extended by Yao *et al.* [39] to obtain FS in the identity-based setting. More precisely, they came up with a forward-secure HIBE (FS-HIBE) constructed via a "cross-product" combination of two HIBE schemes, in the random oracle model. Boneh, Boyen, and Goh [5] offered more efficient FS-HIBE constructions, with selective security in the standard model and with adaptive security in the random oracle model. The first adaptively secure FS-HIBE scheme in the standard model is due to Lewko and Waters [26]. As mentioned by Boyen and Waters [10] and also explored in [13, 16, 33, 34, 37] FS is also achievable for *anonymous* HIBE systems, whose ciphertexts hide the (hierarchy of) identities for which messages were encrypted. Since HIBE generalizes IBE (anonymous) FS-HIBE covers (anonymous) FS-IBE.

FORWARD SECURITY IN ABE/PE. A message encrypted with an ABE/PE scheme can potentially be decrypted by many users. Exposure of some user's private key in these schemes is likely to cause more damage in comparison to PKE or IBE schemes since the adversary could obtain messages that were encrypted for more than one user. Adding forward security to ABE/PE schemes is thus desirable to alleviate this problem. A naïve approach, i.e., to change all keys (incl. public ones) for each new time period, has already been ruled out as being impractical in PKE and IBE schemes, and it seems even more complicated in the ABE/PE setting. In this work we formalize and construct the first forward-secure hierarchical predicate encryption (FS-HPE). Since HPE includes PE/ABE [25, 27], our FS-HPE scheme also implies constructions of first forward secure ABE/PE schemes.

Although forward-secure HIBE constructions exist, formalizing and designing FS-HPE is challenging due to a number of advanced properties that must be considered. In HPE schemes predicates (and by this indirectly private keys) are organized in a hierarchy — any ciphertext that can be decrypted by a low-level predicate must also be decryptable by a high-level predicate but the converse may not be true. In contrast to HIBE, where delegation is performed by extending the parent identity with a substring, predicates in HPE have more complex structures and their delegation requires different techniques. Moreover, predicates should be delegatable at any period in time, irrespective of time evolution for FS. Another aspect is that encryption of messages in forward-secure HPE must be possible only using the master public key, the set of attributes, and the current time period, without having á priori knowledge of predicates at any level of the hierarchy, whereas in FS-HIBE schemes encryption is performed with respect to a given identity at one of the hierarchy levels. As we will discuss in Section 1.2, obtaining forward security in HPE schemes by applying techniques from existing FS-PKE [11] and FS-HIBE [39] results in a number of obstacles. For example, a "cross-product" combination of two HPE schemes [25, 27], akin to the case of two HIBE schemes for FS-HIBE in [39], seems not feasible due to the unique delegation and randomization mechanisms used in those HPE schemes.

Finally, an FS-HPE scheme should still provide attribute-hiding, which could be threatened if (public) time periods for FS are mixed up with attributes during the encryption.

APPLICATIONS. Anonymous (H)IBE found applications in *searchable encryption*: anonymous IBE is known to imply public key encryption with keyword search (PEKS) [1, 7]; anonymous HIBE, as discussed in [1], provides further properties such as temporary PEKS where searching functionality can be restricted to time intervals, and enables new constructions such as identity-based encryption with keyword search (IBEKS). Okamoto and Takashima [27] anticipated HPE applications in the area of *attribute-based searchable encryption*, in particular they mentioned that a two-level HPE can be used to enable attribute-based searchable encryption with predicates (as a generalization of keyword search). We observe that while existing anonymous FS-(H)IBE constructions readily offer forward security to PEKS and IBEKS, obtaining this property for attribute-based searchable encryption would still require a forward secure version of HPE.

According to [10], another application of anonymous (H)IBE is to enable *anonymous and untraceable identity-based encrypted communication* [10], in a similar fashion to key-private public-key encrypted communication mentioned in [2]. We notice that FS property here is crucial, should those privacy goals be preserved for past communications after the exposure of receivers' private keys (which comes close to forward secrecy in key establishment protocols). With (H)PE these privacy properties can be adopted to attribute-based encrypted communication and — if (H)PE schemes are forward secure — remain protected in time periods preceding the exposure of private keys.

1.1 Our Contributions

FS-HPE: MODEL AND SCHEME. We formalize and design the first forward-secure hierarchical predicate encryption (FS-HPE) scheme, for zero-inner-product predicates [24]. Our scheme is secure (adaptively attribute-hiding) in the standard model under the well-known Decision Linear (DLIN) assumption [6] in bilinear groups of prime order. We first present a new syntax and security definitions that are specific to FS-HPE, in particular definition of attribute hiding had to be extended in order to account for FS, in a more complex way than in FS-HIBE definitions from [26, 39], as explained in Section 3.3. Our FS-HPE scheme offers some desirable properties: time-independent delegation of predicates (to support dynamic behavior for delegation of decrypting rights to new users), local update for users' private keys (i.e., no master authority needs to be contacted), forward security, and the scheme's encryption process doesn't require knowledge of predicates at any level including when those predicates join the hierarchy. Considering the relationships amongst the encryption flavors, we can restrict our scheme to level-1 hierarchy and obtain first adaptively-secure FS-PE/ABE construction, or we can set the inner-product predicate to perform the equality test, in which case we would obtain the first adaptively-secure anonymous FS-HIBE scheme under the basic DLIN assumption (as an alternative to [13] that works in bilinear groups of composite order and requires new hardness assumptions).

TECHNIQUES. Our FS-HPE scheme is built based on the dual system encryption approach introduced by Waters [38] and uses the concept of dual pairing vector spaces (DPVS) of Okamoto and Takashima [27]. Techniques underlying forward security of the scheme can be seen as reminiscent of binary tree encryption [11] that was invented for FS-PKE and doesn't apply immediately to the more complex HPE setting. We had to resort to those techniques and modify them for integration with HPE since obtaining FS-HPE in a more direct way, e.g. by adopting the "cross-product" idea from [39], seems not feasible with existing HPE constructions [25, 27]. On a high level, we modify the existing HPE scheme from [25] and combine two of its instances in a non-trivial way to achieve a FS-HPE scheme. One of the HPE schemes handles predicate/attibute hierarchy while another one is used for maintaining time periods using the concept behind binary tree encryption [11]. The modification of the scheme in [25] is necessary to prove security the stringent security definitions involving FS. The combination of two schemes is non-trivial due to the delegation and randomization components inherited from HPE. Our scheme perfectly synchronizes all private key components (decryption, delegation and randomization) from both HPE instances. These components are updated at each new time period and they are also used for time-independent delegation of predicates. We apply game-hopping proofs, following the general proof strategy from [28], i.e. we first define several hard problems and prove that security of our scheme relies on them, then we prove that those hard problems can individually be used to solve the DLIN problem.

1.2 Initial Attempts, or Why FS-HPE is Challenging?

We first discuss several initial attempts to construct FS-HPE and illustrate why combining HPE [25, 27] with FS techniques from [11] is far from being trivial. Informally, in HPE [25, 27] for inner-product relation, hierarchical attributes are represented by $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ and hierarchical predicate vectors are defined as $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ such that $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = 1$ iff $l \leq h$ and $\overrightarrow{x}_i \cdot \overrightarrow{y}_i = 0$ for $1 \leq i \leq l$.

FIRST ATTEMPT. Consider the following na'ive combination of an FS-PKE scheme and an HPE scheme. A private key $SK_{t,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$ of each node in the hierarchy contains two independent secret components: a time-dependent key sk_t for the current time t and a predicate-dependent key $sk_{\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l}$ associated with some hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. Upon delegation, the time-dependent key sk_t is handed over to the user down the hierarchy, in addition to the delegated predicate-dependent keys such that each user can refresh its sk_t autonomously for the next period $t+1$. An HPE ciphertext for time t is created such that it can only be decrypted by an user whose private key contains valid sk_t and a suitable predicate-dependent component. For example, one could first produce an HPE ciphertext c' and then encrypt c' for time t using FS-PKE, to obtain the resulting ciphertext c.

This approach, however, is not forward secure. Assume c FS-encrypts c' for time t where c' is decryptable with $sk_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$. Since time-dependent keys sk_t are shared, at time t the adversary could corrupt some $SK_{t,(\overrightarrow{x}^*_1,\ldots,\overrightarrow{x}^*_l)}$ whose predicate-dependent component $sk_{(\overrightarrow{x}^*_1,\ldots,\overrightarrow{x}^*_l)}$ is not suitable to decrypt c' — this doesn't contradict the intuitive goals behind a forward-secure HPE. Then, at time $t+1$ the adversary corrupts $SK_{t+1,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$. Given sk_t from the first corruption and $sk_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$ from the second corruption the adversary can immediately decrypt the message from c and c'.

SECOND ATTEMPT. Consider the following intuitive modification of the HPE scheme, e.g. [25]. The user with predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ maintains two subtrees with the same root labeled as $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$: the time subtree that evolves over time for forward security (using the concept from [11]), and the predicate subtree to which, in case of delegation, further children can be added to expand the hierarchy. The time subtree can be viewed as a special predicate subtree where each time node ID_t is denoted by a two dimensional predicate vector [24], for example $\overrightarrow{x}_{ID_t} = (-ID_t, 1)$. To encrypt a message at time t, the sender uses attribute vectors $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h, \overrightarrow{y}_{ID_t})$, where ID_t denotes the node for time t and $\overrightarrow{y}_{ID_t} = (1, ID_t)$. This message can be decrypted, provided that the receiver knows the secret key for time t and the ciphertext's attributes $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ satisfy his private key predicates $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$.

The limitation of this approach is that key delegation is not independent of the time period. To guarantee forward security the user must erase the secret key corresponding to the root $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$, as its exposure would compromise the secrecy of derived keys for earlier time periods. However, if this secret key is erased then the predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ can no longer be expanded, i.e. the scheme will not be able to support hierarchical key delegation at the level of that predicate. This attempts indicates that all secret keys must evolve together.

THIRD ATTEMPT. Consider a more direct FS-HPE construction where all private keys evolve over the time. Assume that attribute vectors (resp. predicate vectors) consist of alternating attributes (resp. predicates) and time identifiers, which is referred to as an attribute-time-tuple (resp. predicate-time-tuple). The private key at each node serves three purposes: decryption, hierarchical delegation, and derivation of private keys for the next time period. Predicate vectors associated with the private key of a newly added node are defined by predicate-time-tuple of its parent extended with the node-specific predicate vector. This private key can in turn be used to derive further keys for its descendants. For example, if $\overrightarrow{x}_2$ extends the root $\overrightarrow{x}_1$ at time t_1 and $\overrightarrow{x}_3$ extends $\overrightarrow{x}_2$ at time t_2, the hierarchical predicate for the third node at time t_2 is given by $(\overrightarrow{x}_1, \overrightarrow{x}_{ID_{t_1}}, \overrightarrow{x}_2, \overrightarrow{x}_{ID_{t_2}}, \overrightarrow{x}_3)$, where $\overrightarrow{x}_{ID_{t_1}} = (-ID_{t_1}, 1)$ and $\overrightarrow{x}_{ID_{t_2}} = (-ID_{t_2}, 1)$ (as in the previous attempt).

In order to decrypt a message using this hierarchical predicate the ciphertext must contain attributes for the time periods t_1 and t_2. That is, the sender must know all time periods at which different nodes on the path joined the hierarchy. Clearly, this is a limitation in terms of both scalability and privacy. That is, different private keys within the predicate hierarchy should ideally evolve in time that remains transparent to the encryption algorithm.

Our exposition above shows that designing FS-HPE is not straightforward eventhough the forward security concept has been known in other encryption flavors. In our scheme, presented in Section 4, that is based on a variant of the HPE

scheme from [25] and a reminiscent of the binary tree encryption from [11], included additional tricks to overcome problems demonstrated above.

2 Dual Pairing Vector Spaces and Assumptions

GROUPS. Let $\mathcal{G}_{\mathsf{bpg}}$ be an algorithm that on input a security parameter 1^λ outputs a description of the symmetric bilinear group setting $(q, \mathbb{G}, \mathbb{G}_T, G, e)$ where q is a prime, $\mathbb{G}$ and $\mathbb{G}_T$ are two cyclic groups of order q, G is the generator of $\mathbb{G}$, e is a non-degenerate bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, i.e., $e(sG, tG) = e(G, G)^{st}$ and $e(G, G) \neq 1$. We also define cyclic additive group $\mathbb{G}$ and multiplicative group $\mathbb{G}_T$ of order q.

VECTOR SPACES. Let $\mathbb{V} = \overbrace{\mathbb{G} \times \cdots \times \mathbb{G}}^{N}$ be a *vector space* and each element in $\mathbb{V}$ be expressed by *N-dimensional vector*. $\boldsymbol{x} = (x_1 G, \ldots, x_N G)$ ($x_i \in \mathbb{F}_q$ for $i = 1, \ldots, N$). The *canonical base* $\mathbb{A}$ of $\mathbb{V}$ is $\mathbb{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$, where $\boldsymbol{a}_1 = (G, 0, \ldots, 0)$, $\boldsymbol{a}_2 = (0, G, 0, \ldots, 0), \ldots, \boldsymbol{a}_N = (0, \ldots, 0, G)$. Given two vectors $\boldsymbol{x} = (x_1 G, \ldots, x_N G) = x_1 \boldsymbol{a}_1 + \cdots + x_N \boldsymbol{a}_N \in \mathbb{V}$ and $\boldsymbol{y} = (y_1 G, \ldots, y_N G) = y_1 \boldsymbol{a}_1 + \cdots + y_N \boldsymbol{a}_N \in \mathbb{V}$, where $\overrightarrow{x} = (x_1, \ldots, x_N)$ and $\overrightarrow{y} = (y_1, \ldots, y_N)$, the pairing operation is defined as $e(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} e(x_i G, y_i G) = e(G, G)^{\sum_{i=1}^{N} x_i y_i} = g_T^{\overrightarrow{x}\,\overrightarrow{y}} \in \mathbb{G}_T$.

Definition 1 (Dual Pairing Vector Space (DPVS) [27]). *Let $(q, \mathbb{G}, \mathbb{G}_T, G, e)$ be a symmetric bilinear pairing group. A* Dual Pairing Vector Space *$(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$, generated by an algorithm denoted $\mathcal{G}_{\mathsf{dpvs}}$, is a tuple containing a prime q, an N-dimensional vector space $\mathbb{V}$ over $\mathbb{F}_q$, a cyclic group $\mathbb{G}_T$ of order q, a canonical base $\mathbb{A} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$ of $\mathbb{V}$, and a pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ that satisfy the following conditions:*

1. NON-DEGENERATE BILINEAR PAIRING: *There exists a polynomial-time computable non-degenerate bilinear pairing $e(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{N} e(G_i, H_i)$ where $\boldsymbol{x} = (G_1, \ldots, G_N) \in \mathbb{V}$ and $\boldsymbol{y} = (H_1, \ldots, H_N) \in \mathbb{V}$. This is non-degenerate bilinear pairing i.e., $e(s\boldsymbol{x}, t\boldsymbol{y}) = e(\boldsymbol{x}, \boldsymbol{y})^{st}$ and if $e(\boldsymbol{x}, \boldsymbol{y}) = 1$ for all $\boldsymbol{y} \in \mathbb{V}$, then $\boldsymbol{x} = \boldsymbol{0}$.*
2. DUAL ORTHONORMAL BASES: *$\mathbb{A}$ and e satisfy that $e(\boldsymbol{a_i}, \boldsymbol{a_j}) = g_T^{\delta_{i,j}}$ for all i and j, where $\delta_{i,j} = 1$ if $i = j$, and 0 otherwise, and $g_T \neq 1 \in \mathbb{G}_T$.*
3. DISTORTION MAPS: *Linear transformations $\phi_{i,j}$ on $\mathbb{V}$ s.t. $\phi_{i,j}(\boldsymbol{a}_j) = \boldsymbol{a}_i$ and $\phi_{i,j}(\boldsymbol{a}_k) = \boldsymbol{0}$ if $k \neq j$ are polynomial-time computable. We call $\phi_{i,j}$ "distortion maps".*

ORTHONORMAL BASES. Let $\mathbb{B} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$ be a basis of vector space $\mathbb{V}$ which is obtained from its canonical basis $\mathbb{A}$ using a uniformly chosen linear transformation $\Lambda = (\lambda_{i,j}) \xleftarrow{U} GL(N, \mathbb{F}_q)$. Note that $GL(N, \mathbb{F}_q)$ creates a matrix of size $N \times N$ in which each element is uniformly selected from $\mathbb{F}_q$ such that $\boldsymbol{b}_i = \sum_{j=1}^{N} \lambda_{i,j} \boldsymbol{a}_j$, for $i = 1, \ldots, N$. Similarly, let $\mathbb{B}^* = (\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_N^*)$ be another basis of $\mathbb{V}$ which is also obtained from $\mathbb{A}$ using $\mu_{i,j} = (\Lambda^T)^{-1}$ as $\boldsymbol{b}_i^* = \sum_{j=1}^{N} \mu_{i,j} \boldsymbol{a}_j$, for $i = 1, \ldots, N$. It can be shown that $e(\boldsymbol{b}_i, \boldsymbol{b}_j^*) = g_T^{\delta_{i,j}}$, where $\delta_{i,j} = 1$ if $i = j$, and $\delta_{i,j} = 0$ if $i \neq j$. That is $\mathbb{B}$ and $\mathbb{B}^*$ are dual orthonormal bases of $\mathbb{V}$. In our scheme we will use the following probabilistic algorithm $\mathcal{G}_{\mathsf{ob}}$ to generate group and DPSV parameters and the two dual orthonormal bases:

$$
\begin{aligned}
\mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n} = (d; n_1, \ldots, n_d)) : \mathsf{param}_{\mathbb{G}} &= (q, \mathbb{G}, \mathbb{G}_T, G, e) \xleftarrow{R} \mathcal{G}_{\mathsf{bpg}}(1^\lambda), \\
&\psi \xleftarrow{U} \mathbb{F}_q^\times, N_0 = 5, N_t = 3n_t + 1 \text{ for } t = 1, \ldots, d; \\
&\text{For } t = 0, \ldots, d : \\
&\quad \mathsf{param}_{\mathbb{V}_t} = (q, \mathbb{V}_t, \mathbb{G}_T, \mathbb{A}_t, e) \xleftarrow{R} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N_t, \mathsf{param}_{\mathbb{G}}), \\
&\quad \Lambda^{(t)} = (\lambda_{i,j}^{(t)}) \xleftarrow{U} GL(N_t, \mathbb{F}_q), \ (\mu_{i,j}^{(t)}) = \psi \cdot (\Lambda^{(t)^T})^{-1}, \\
&\quad \boldsymbol{b}_i{}^{(t)} = \sum_{j=1}^{N_t} \lambda_{i,j}^{(t)} \boldsymbol{a}_j^{(t)} \text{ for } i = 1, \ldots, N_t, \mathbb{B}^{(t)} = (\boldsymbol{b}_1^{(t)}, \ldots, \boldsymbol{b}_{N_t}^{(t)}),
\end{aligned}
$$

$$\boldsymbol{b}_i^{*(t)} = \sum_{j=1}^{N_t} \mu_{i,j}^{(t)} \boldsymbol{a}_j^{(t)} \text{ for } i = 1, \ldots, N_t, \mathbb{B}^{*(t)} = (\boldsymbol{b}_1^{*(t)}, \ldots, \boldsymbol{b}_{N_t}^{*(t)}),$$
$$g_T = e(G,G)^{\psi}, \mathsf{param}_{\overrightarrow{n}} = (\{\mathsf{param}_{\mathbb{V}_t}\}_{t=0,\ldots,d}, g_T),$$
$$\text{Output } (\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(t)}, \mathbb{B}^{*(t)}\}_{t=0,\ldots,d}).$$

Note that $g_T = e(\boldsymbol{b}_i{}^{(t)}, \boldsymbol{b}_i^{*(t)})$ for $t = 0, \ldots, d$; $i = 1, \ldots, N_t$.

Definition 2 (Decisional Linear Assumption (DLIN) [6]). *The DLIN problem is to find bit $\beta \in \{0,1\}$, given the output $(\mathsf{param}_{\mathbb{G}}, G, aG, bG, acG, bdG, Y_\beta)$ of the probabilistic algorithm*

$$\mathcal{G}_\beta^{\mathrm{DLIN}}(1^\lambda) : \mathsf{param}_{\mathbb{G}} = (q, \mathbb{G}, \mathbb{G}_T, G, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bpg}}(1^\lambda), a, b, c, d \xleftarrow{\mathsf{U}} \mathbb{F}_q,$$
$$Y_0 = (c+d)G,\ Y_1 \xleftarrow{\mathsf{U}} \mathbb{G},\ \beta \xleftarrow{\mathsf{U}} \{0,1\};$$
$$\text{Output } (\mathsf{param}_{\mathbb{G}}, G, aG, bG, acG, bdG, Y_\beta).$$

The advantage of a probabilistic polynomial-time DLIN solver $\mathcal{D}$ is defined as

$$\mathsf{Adv}_{\mathcal{D}}^{\mathrm{DLIN}}(\lambda) = \left| \mathsf{Pr}\left[\mathcal{D}(1^\lambda, \varpi) \to 1 \,\middle|\, \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathrm{DLIN}}(1^\lambda)\right] - \mathsf{Pr}\left[\mathcal{D}(1^\lambda, \varpi) \to 1 \,\middle|\, \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathrm{DLIN}}(1^\lambda)\right] \right|.$$

The DLIN assumption states that for any such $\mathcal{D}$ this advantage is negligible in λ.

3 Forward-Secure Hierarchical Predicate Encryption

In this section we present our model for forward secure hierarchical predicate encryption (FS-HPE). First, we highlight the idea behind FS-HPE concept and introduce some notations. In FS-HPE private keys are associated with predicate vectors and evolve over the time. At any time period i a user may join the hierarchy and receive delegated private keys. These keys are computed by the parent user for time period i and together with further secret information that is necessary to derive private keys for later time periods is handed over to the joined user. Once the user receives this secret information, at the end of each period the user updates his private key locally and erases secrets that are no longer needed. Additionally, at any time $j \geq i$ the user may delegate its private key down the hierarchy without contacting its parent. In any time period i a message can be encrypted using public parameters, the attribute vectors, and i. In order to decrypt for time period i users must possess private keys satisfying attributes from the ciphertext for that time.

3.1 Notations

Time Period Let the total number of time periods $N = 2^\kappa$, where $\kappa \in \mathbb{N}$.

Hierarchical Inner-Product Predicate Encryption We borrow some notations from [25] to describe our HPE with inner-product predicates (in Appendix A we also recall the syntax and the HPE scheme from [25]). Let $\overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)$ be a tuple of positive integers such that $\mu_0 = 0 < \mu_1 < \mu_2 < \cdots < \mu_d = n$. We call $\overrightarrow{\mu}$ a *format of hierarchy of depth d attribute spaces*. With Σ_l, $l = 1, \ldots, d$ we denote *attribute sets* and each $\Sigma_l = \mathbb{F}_q^{\mu_l - \mu_{l-1}} \setminus \{0\}$. A *hierarchical attribute* $\Sigma = \cup_{l=1}^d (\Sigma_1 \times \ldots \times \Sigma_l)$ is defined using the disjoint union. For $\overrightarrow{v}_i \in \mathbb{F}_q^{\mu_i - \mu_{i-1}} \setminus \{\overrightarrow{0}\}$, a hierarchical attribute $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) \in \Sigma$ is said to satisfy a *hierarchical predicate* $f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$ iff $l \leq h$ and $\overrightarrow{x}_i \cdot \overrightarrow{y}_i = 0$ for $1 \leq i \leq l$, which we denote as $f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = 1$. The space of hierarchical predicates is $\mathcal{F} = \{f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)} | \overrightarrow{x}_i \in \mathbb{F}_q^{\mu_i - \mu_{i-1}} \setminus \{\overrightarrow{0}\}\}$. We call h (resp. l) the *level* of $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ (resp. $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$). Throughout the paper we will assume that an attribute vector $\overrightarrow{y}_1 = (y_1, \ldots, y_{\mu_1})$ is normalized such that $y_1 = 1$ (note that $\overrightarrow{y}_1$ can be normalized via $(1/y_1) \cdot \overrightarrow{y}_1$, assuming that y_1 is non-zero). By $\overrightarrow{e}_i^{(k)}$ we denote the *canonical basis vector* $(\overbrace{0, \ldots, 0}^{i-1}, 1, \overbrace{0, \ldots, 0}^{n_k - i}) \in \mathbb{F}_q^{n_k}$ for $k = 1, 2$ and $i = 1, \ldots, n_k$.

Keys We will work with two notations for secret keys: $sk_{w,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$ is the key associated with some prefix w of the bit representation of a time period i and a hierarchical predicate $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$, whereas $SK_{i,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$ denotes the key associated with time i and a hierarchical predicate $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$. That is, $SK_{i,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)} = \{sk_{i,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}, sk_{w1,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)} : w0 \text{ is a prefix of } i\}$.

3.2 Syntax

Definition 3. *A forward secure hierarchical predicate encryption scheme* (FS-HPE) *is a tuple of five algorithms* (RootSetup, Delegate, Update, Encrypt, Decrypt) *described in the following:*

- RootSetup$(1^\lambda, N, \overrightarrow{\mu})$ *This algorithm takes as input a security parameter 1^λ, the total number of time periods N and the format of hierarchy $\overrightarrow{\mu}$. It outputs public parameters of the system, incl. public key PK, and a root secret key $SK_{0,1}$, which is assumed to be known only to the master authority of the hierarchy.*
- Delegate$(SK_{i,l}, i, \overrightarrow{x}_{l+1})$ *This algorithm takes as input a secret key $SK_{i,l}$ associated with time i on hierarchy level l and an $(l+1)$-th level predicate vector $\overrightarrow{x}_{l+1}$. It outputs the delegated secret key $SK_{i,l+1}$. This key is intended for the direct descendant at level $l+1$. It is assumed that predicate vector $\overrightarrow{x}_{l+1}$ is added to the predicate hierarchy during the time period i.*
- Update$(SK_{i,l}, i)$ *This algorithm takes as input a secret key $SK_{i,l}$ and the current time period i. It outputs an updated secret key $SK_{i+1,l}$ for the following time period $i+1$ and erases $SK_{i,l}$.*
- Encrypt$(PK, (\overrightarrow{y}_1,\ldots,\overrightarrow{y}_h), i, M)$ *This algorithm takes as input the public key PK, hierarchical attribute vectors $(\overrightarrow{y}_1,\ldots,\overrightarrow{y}_h)$, a time period i, and a message M from the associated message space. It outputs a ciphertext C. We assume that i is included in C.*
- Decrypt$(C, SK_{i,l})$ *This algorithm takes as input a ciphertext C and a secret key $SK_{i,l}$ for the time period i and predicate vectors $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$. It outputs either a message M or the distinguished symbol $\perp$ (to indicate a failure).*

Correctness. *For all correctly generated PK and $SK_{i,l}$ associated with predicate vectors $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$ and a time period i, let $C \xleftarrow{R}$ Encrypt$(PK, (\overrightarrow{y}_1,\ldots,\overrightarrow{y}_h), i, M)$ and $M' =$ Decrypt$(C, SK_{i,l})$. Then, if $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1,\ldots,\overrightarrow{y}_h) = 1$ then $M = M'$; otherwise, $M \neq M'$ with all but negligible probability.*

3.3 Security Definition

Definition 4. *A* FS-HPE *scheme is* adaptively attribute hiding against chosen plaintext attacks *if for all PPT adversaries $\mathcal{A}$, the advantage of $\mathcal{A}$ in the following game is negligible in the security parameter:*

Setup. RootSetup *algorithm is run by the challenger $\mathcal{C}$ to generate public key PK and root secret key $SK_{0,1}$. PK is given to $\mathcal{A}$.*

Queries I. *$\mathcal{A}$ may adaptively make a polynomial number of* delegation queries *by asking $\mathcal{C}$ to create a secret key for any given time period i and hierarchical predicate vectors $(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)$. In response, $\mathcal{C}$ computes the secret key $SK_{i,l}$ and reveals it to $\mathcal{A}$. (Note that $\mathcal{C}$ computes $SK_{i,l}$ with the help of algorithms* Delegate *and* Update *that it may need to execute several times, i.e. depending on the input time period i and hierarchy level l.)*

Challenge. *$\mathcal{A}$ outputs its challenge, containing two attribute vectors $(Y^{(0)}, Y^{(1)}) = ((\overrightarrow{y}_1^{(0)},\ldots,\overrightarrow{y}_{h^{(0)}}^{(0)}), (\overrightarrow{y}_1^{(1)},\ldots,\overrightarrow{y}_{h^{(1)}}^{(1)}))$, two plaintexts $(M^{(0)}, M^{(1)})$, and a time period I, such that*

- *either $i > I$, or*
- *$i \leq I$ and $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1^{(0)},\ldots,\overrightarrow{y}_{h^{(0)}}^{(0)}) = f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1^{(1)},\ldots,\overrightarrow{y}_{h^{(1)}}^{(1)}) = 0$,*

for each revealed key for $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$ and time period i. $\mathcal{C}$ then flips a random coin b. If $b = 0$ then $\mathcal{A}$ is given $C =$ Encrypt$(PK, Y^{(0)}, I, M^{(0)})$ and if $b = 1$ then $\mathcal{A}$ is given $C =$ Encrypt$(PK, Y^{(1)}, I, M^{(1)})$.

Query phase 2. *Repeat the* **Query phase 1** *subject to the restrictions as in the challenge phase.*

Guess. *$\mathcal{A}$ outputs a bit b', and succeeds if $b' = b$.*

We define the advantage of $\mathcal{A}$ as a quantity $\mathsf{Adv}_{\mathcal{A}}^{\text{FS-HPE}}(\lambda) = |\Pr[b = b'] - 1/2|$.

Remark 1. In Definition 4, adversary $\mathcal{A}$ is not allowed to ask a key query for time period i and hierarchical predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ such that $i \leq I$ and $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)}) = 1$ for some $b \in \{0,1\}$, i.e., the queried key is not allowed to decrypt the challenge ciphertext. Recently, Okamoto and Takashima [31] proposed a PE (HPE) which allow such key query, provided that $M^{(0)} = M^{(1)}$. The technique of Okamoto and Takashima [31] can be applied in our scheme to achieve strong security.

Remark 2. In Definition 4, $\mathcal{A}$ may ask delegation queries and obtain the resulting keys. This contrasts slightly with the HPE security definition in [25], where $\mathcal{A}$ may ask the challenger to create and delegate private keys but will not be given any of them, unless it explicitly asks a separate reveal query. This is because HPE in [25] has two algorithms for computing secret keys, either directly (using the master secret key) or through delegation (using secret key of the parent node). In our FS-HPE syntax we compute secret keys through delegation only and in the security definition we are mainly concerned with maintaining time evolution for delegated keys.

Remark 3. Definition 4 can be easily extended to address chosen-ciphertext attacks (CCA) by allowing decryption queries. The usual restriction is that decryption queries cannot be used for the challenge ciphertext. Our CPA-secure FS-HPE scheme from Section 4 can be strengthened to resist CCA by applying the well-known CHK transformation from [12] that uses one-time signatures to authenticate the ciphertext.

4 Our Forward-Secure HPE Scheme

4.1 High-Level Description

For simplicity of presentation, our FS-HPE makes use of a version of FS-PKE scheme by Katz [22]. In Katz's scheme, time periods are associated with the leaf nodes of a binary tree while in Canetti *et al.* scheme [11], time periods correspond to all nodes of the tree. Our scheme can also be realized based on the FS-PKE scheme by Canetti et al., which will give faster key update time. We utilize a full binary tree of height κ, whose root is labeled ϵ and all other nodes are labeled recursively: if the label of a node is w, then its left child is $w0$, and its right child is $w1$. Each time period $i \in \{0, \ldots, N-1\}$ corresponds to a leaf identified via the binary representation of i. We denote the k-bit prefix of a d-length word $w = w_1 w_2 \ldots w_d$ by $w|_k$, i.e. $w|_k = w_1 w_2 \ldots w_k$ for $k \leq d$. Let $w|_0 = \epsilon$ and $w = w|_d$.

We use two HPE schemes in parallel. Private keys in each scheme contain three components: decryption, delegation and randomness. Private key of a user contains private keys from both schemes that are linked together using secret sharing. One HPE scheme is used to handle predicate/attribute hierarchy, while the other one is used to handle time evolution. Each of the two HPE schemes is a modification of the scheme in [25], in a way that allows us to prove attribute-hiding property under more sophisticated conditions involving time evolution. The efficiency of the modified scheme is still comparable to the one in [25], i.e. it increases the ciphertext by an additional component (master component) that is used to combine both HPE schemes and is crucial for the security proof. This change implies that the length of the orthonormal bases grows from $(2n+3) \cdot |G|$ in [25] to $(3n+1) \cdot |G|$ in our scheme, where n is the dimension of the attribute vectors, and $|G|$ is the length of a group element from G.

At time period i, the entity at level l with a hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ holds a secret key $SK_{i,(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}$, denoted for simplicity as $SK_{i,l}$. It contains secret keys $sk_{i,l}$ and $\{sk_{w,l}\}$) for each label w corresponding to a right sibling node (if one exists) on the path from l to the root. We view $sk_{i,l}$ as a decryption key, which is associated with current time i and the predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. The secret keys in $\{sk_{w,l}\}$ contain auxiliary information used to update $sk_{i,l}$ for future time periods and to derive its lower-level predicates. The initial keys $sk_{0,1}$ and $sk_{1,1}$ are computed in the RootSetup algorithm and are associated with the predicate $\overrightarrow{x}_1$. In general, each $sk_{w,l}$ contains three secret components: the *decryption component* $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}})$, the *randomness component* $(\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},|w|+1})$ and the *delegation component* $(\boldsymbol{k}^{(1)}_{w,l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2|w|+1}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},L})$. All above components are constructed using orthonormal bases $\mathbb{B}^*$ specified in Section 2. There are three different bases in the system. The superscript of each key component denotes its base. $\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}$ is the mentioned master component that links $\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}$ and $\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}}$ using the secret sharing techniques. In turn, $\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}$ and

$\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}}$ are used in respective HPE schemes. If w represents a leaf of the binary tree then the decryption component $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}})$ is used for decryption at time represented by w.

Delegation and randomization of private keys are processed similarly as in [25], except that upon derivation of keys for lower level predicates, we also delegate and randomize their time-dependent part. In particular, the delegation component of the l-th level key is essential to compute the $(l+1)$-th level child key, and the randomness component of the l-th level key is used to re-randomize the latter's coefficients. To handle time hierarchy we deploy "dummy" nodes. Similarly, we will compute the dummy child for predicate hierarchy when time evolves. In this way, all derived keys are re-randomized.

We define a helper algorithm ComputeNext that will be called from RootSetup and Update. Given a secret key $sk_{w,l}$ for node w and a hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ it outputs $sk_{(wb),l}$, $b \in \{0,1\}$ for the nodes $w0$ and $w1$ by updating the three components of $sk_{w,l}$. The algorithm Update computes secret keys for the next time period through the internal call to ComputeNext and erases all secret information that was used to derive the key for the current time period. The update procedure involves all three components of the secret key. For example, for a given secret key $SK_{i,l} = (sk_{i,l}, \{sk_{w,l}\})$, forward security is achieved by deleting its component $sk_{i,l}$ and using all three components of $\{sk_{w,l}\}$, where w is now the label of an internal node, to derive $SK_{i+1,l}$ for the following time period with the help of ComputeNext.

In algorithm Delegate, a secret key $sk_{w,l}$ for a string w is used to derive $sk_{w,u}$ for a lower hierarchy level $u > l$ and a hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_u)$ that has restricted capabilities in comparison to $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. As mentioned, the delegation component for hierarchical predicates of $sk_{w,l}$ is essential for the derivation of $sk_{w,u}$, whose coefficients are re-randomized with the randomization component.

The algorithm Encrypt requires only a time period t and a hierarchical attribute $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ to encrypt the message. We note that during encryption attributes $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ are extended with random elements from level $h+1$ down to the leaf, i.e., the scheme encrypts attribute vectors on all levels in the hierarchy instead of encrypting only the input vectors. In this way, parent keys can directly decrypt ciphertexts produced for their children without taking effort to derive child keys first.

The algorithm Decrypt uses the decryption key $sk_{i,l}$, which is associated with time period i and hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. The message is decrypted iff the attributes in the ciphertext satisfy the predicates in the decryption component of the key and the ciphertext is created at time i.

Remark 4. The two HPE schemes underlying our construction are modification of the HPE scheme from [25]. We observe, however, that the recent (H)PE schemes of Okamoto and Takashima [29, 30] could also be modified (in a similar way as [25]) to build a forward secure HPE scheme using our techniques. In this case the resulting FS-HPE scheme would have shorter private keys than in our current construction.

4.2 Detailed Specification

The five algorithms of our FS-HPE scheme are detailed in the following:

Algorithm RootSetup$\big(1^\lambda, N = 2^\kappa, \overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)\big)$: Let $\overrightarrow{x}_1$ be the root predicate and let $L = 2\kappa$ and $\overrightarrow{n} = (2; n, L)$. This initialization algorithm proceeds as follows.

Compute

$$(\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{*(2)}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}),$$

$$\widetilde{\mathbb{B}}^{(0)} = (\boldsymbol{b}^{(0)}_1, \boldsymbol{b}^{(0)}_3, \boldsymbol{b}^{(0)}_5), \widetilde{\mathbb{B}}^{(1)} = (\boldsymbol{b}^{(1)}_1, \ldots, \boldsymbol{b}^{(1)}_n, \boldsymbol{b}^{(1)}_{3n+1}), \widetilde{\mathbb{B}}^{(2)} = (\boldsymbol{b}^{(2)}_1, \ldots, \boldsymbol{b}^{(2)}_L, \boldsymbol{b}^{(2)}_{3L+1}),$$

$$\widetilde{\mathbb{B}}^{*(0)} = (\boldsymbol{b}^{*(0)}_1, \boldsymbol{b}^{*(0)}_3), \widetilde{\mathbb{B}}^{*(1)} = (\boldsymbol{b}^{*(1)}_1, \ldots, \boldsymbol{b}^{*(1)}_n), \widetilde{\mathbb{B}}^{*(2)} = (\boldsymbol{b}^{*(2)}_1, \ldots, \boldsymbol{b}^{*(2)}_L),$$

$$\widehat{\mathbb{B}}^{*(1)} = (\boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n}), \widehat{\mathbb{B}}^{*(2)} = (\boldsymbol{b}^{*(2)}_{2L+1}, \ldots, \boldsymbol{b}^{*(2)}_{3L}).$$

The master authority needs to generate not only the secret key associated with the current time period 0 but also secret keys corresponding to the internal nodes on the binary tree whose bit representations are all 0 except for the last bit. The secret key for time 0 and predicate $\overrightarrow{x}_1$ is denoted as $sk_{0^\kappa,1}$. Secret keys that will be used to derive keys for future time periods are denoted as $\{sk_{1,1}, sk_{(01),1}, \ldots, sk_{0^{\kappa-1}1,1}\}$. These values are generated recursively as follows, starting with $sk_{0,1}$ and $sk_{1,1}$.

Computing $sk_{0,1}$:

Pick $\psi, \psi', \alpha_{\mathsf{dec}}, \alpha^{(1)}_{\mathsf{dec}}, \alpha^{(2)}_{\mathsf{dec}} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ such that $\alpha_{\mathsf{dec}} = \alpha^{(1)}_{\mathsf{dec}} + \alpha^{(2)}_{\mathsf{dec}}$.

Pick $\eta^{(0)}_{\mathsf{dec}}, \beta^{(1)}_{\mathsf{dec},1}, \beta^{(2)}_{\mathsf{dec},1}, \beta^{(1)}_{\mathsf{ran},j,1}(j = 1,2), \beta^{(2)}_{\mathsf{ran},j,1}(j = 1,2), \beta^{(1)}_{\mathsf{del},j,1}(j = 1,\ldots,n), \beta^{(2)}_{\mathsf{del},j,1}(j = 1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\overrightarrow{\eta}^{(2)}_{\mathsf{dec}}, \overrightarrow{\eta}^{(2)}_{\mathsf{ran},j}(j = 1,2), \overrightarrow{\eta}^{(2)}_{\mathsf{del},j}(j = 1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q^L$, $\overrightarrow{\eta}^{(1)}_{\mathsf{dec}}, \overrightarrow{\eta}^{(1)}_{\mathsf{ran},j}(j = 1,2), \overrightarrow{\eta}^{(1)}_{\mathsf{del},j}(j = 1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q^n$.

Compute

$$
\begin{aligned}
\boldsymbol{k}^{(0)}_{0,1,\mathsf{dec}} &= (-\alpha_{\mathsf{dec}}, 0, 1, \eta^{(0)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(0)}},\\
\boldsymbol{k}^{(1)}_{0,1,\mathsf{dec}} &= (\alpha^{(1)}_{\mathsf{dec}} \overrightarrow{e}^{(1)}_1 + \beta^{(1)}_{\mathsf{dec},1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\eta}^{(1)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(1)}},\\
\boldsymbol{k}^{(2)}_{0,1,\mathsf{dec}} &= (\alpha^{(2)}_{\mathsf{dec}}, \beta^{(2)}_{\mathsf{dec},1}, 0^{2L-2}, \overrightarrow{\eta}^{(2)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(2)}},\\
\boldsymbol{k}^{(1)}_{0,1,\mathsf{ran},j} &= (\beta^{(1)}_{\mathsf{ran},j,1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\eta}^{(1)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j = 1,2,\\
\boldsymbol{k}^{(2)}_{0,1,\mathsf{ran},j} &= (0, \beta^{(2)}_{\mathsf{ran},j,1}, 0^{2L-2}, \overrightarrow{\eta}^{(2)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j = 1,2,\\
\boldsymbol{k}^{(1)}_{0,1,\mathsf{del},j} &= (\beta^{(1)}_{\mathsf{del},j,1} \overrightarrow{x}_1, 0^{j-\mu_1-1}, \psi, 0^{2n-j}, \overrightarrow{\eta}^{(1)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j = \mu_1+1,\ldots,n,\\
\boldsymbol{k}^{(2)}_{0,1,\mathsf{del},j} &= (0, \beta^{(2)}_{\mathsf{del},j,1}, 0^{j-3}, \psi', 0^{2L-j}, \overrightarrow{\eta}^{(2)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j = 3,\ldots,L.
\end{aligned}
$$

Let $sk_{0,1} = (\boldsymbol{k}^{(0)}_{0,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{ran},1}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{ran},2}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{ran},1}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{ran},2}, \boldsymbol{k}^{(1)}_{0,1,\mathsf{del},\mu_1+1}, \ldots, \boldsymbol{k}^{(1)}_{0,1,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{0,1,\mathsf{del},3}, \ldots, \boldsymbol{k}^{(2)}_{0,1,\mathsf{del},L})$.

Computing $sk_{1,1}$:

Pick $\pi, \pi', \delta_{\mathsf{dec}}, \delta^{(1)}_{\mathsf{dec}}, \delta^{(2)}_{\mathsf{dec}} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ such that $\delta_{\mathsf{dec}} = \delta^{(1)}_{\mathsf{dec}} + \delta^{(2)}_{\mathsf{dec}}$.

Pick $\gamma^{(0)}_{\mathsf{dec}}, \theta^{(1)}_{\mathsf{dec},1}, \theta^{(2)}_{\mathsf{dec},1}, \theta^{(1)}_{\mathsf{ran},j,1}(j = 1,2), \theta^{(2)}_{\mathsf{ran},j,1}(j = 1,2), \theta^{(1)}_{\mathsf{del},j,1}(j = 1,\ldots,n), \theta^{(2)}_{\mathsf{del},j,1}(j = 1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\overrightarrow{\gamma}^{(1)}_{\mathsf{dec}}, \overrightarrow{\gamma}^{(1)}_{\mathsf{ran},j}(j = 1,2), \overrightarrow{\gamma}^{(1)}_{\mathsf{del},j}(j = 1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q^n$, $\overrightarrow{\gamma}^{(2)}_{\mathsf{dec}}, \overrightarrow{\gamma}^{(2)}_{\mathsf{ran},j}(j = 1,2), \overrightarrow{\gamma}^{(2)}_{\mathsf{del},j}(j = 1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q^L$.

Compute

$$
\begin{aligned}
\boldsymbol{k}^{(0)}_{1,1,\mathsf{dec}} &= (-\delta_{\mathsf{dec}}, 0, 1, \gamma^{(0)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(0)}},\\
\boldsymbol{k}^{(1)}_{1,1,\mathsf{dec}} &= (\delta^{(1)}_{\mathsf{dec}} \overrightarrow{e}^{(1)}_1 + \theta^{(1)}_{\mathsf{dec},1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\gamma}^{(1)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(1)}},\\
\boldsymbol{k}^{(2)}_{1,1,\mathsf{dec}} &= (\delta^{(2)}_{\mathsf{dec}} + \theta^{(2)}_{\mathsf{dec},1}, \theta^{(2)}_{\mathsf{dec},1}, 0^{2L-2}, \overrightarrow{\gamma}^{(2)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(2)}},\\
\boldsymbol{k}^{(1)}_{1,1,\mathsf{ran},j} &= (\theta^{(1)}_{\mathsf{ran},j,1} \overrightarrow{x}_1, 0^{2n-\mu_1}, \overrightarrow{\gamma}^{(1)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j = 1,2,\\
\boldsymbol{k}^{(2)}_{1,1,\mathsf{ran},j} &= (\theta^{(2)}_{\mathsf{ran},j,1}, \theta^{(2)}_{\mathsf{ran},j,1}, 0^{2L-2}, \overrightarrow{\gamma}^{(2)}_{\mathsf{ran},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j = 1,2,\\
\boldsymbol{k}^{(1)}_{1,1,\mathsf{del},j} &= (\theta^{(1)}_{\mathsf{del},j,1} \overrightarrow{x}_1, 0^{j-\mu_1-1}, \pi, 0^{2n-j}, \overrightarrow{\gamma}^{(1)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j = \mu_1+1,\ldots,n,\\
\boldsymbol{k}^{(2)}_{1,1,\mathsf{del},j} &= (\theta^{(2)}_{\mathsf{del},j,1}, \theta^{(2)}_{\mathsf{del},j,1}, 0^{j-3}, \pi', 0^{2L-j}, \overrightarrow{\gamma}^{(2)}_{\mathsf{del},j}, 0)_{\mathbb{B}^{*(2)}}, \text{ for } j = 3,\ldots,L.
\end{aligned}
$$

Let $sk_{1,1} = (\boldsymbol{k}^{(0)}_{1,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{ran},1}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{ran},2}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{ran},1}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{ran},2}, \boldsymbol{k}^{(1)}_{1,1,\mathsf{del},\mu_1+1}, \ldots, \boldsymbol{k}^{(1)}_{1,1,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{1,1,\mathsf{del},3}, \ldots, \boldsymbol{k}^{(2)}_{1,1,\mathsf{del},L})$.

Recursion: Use $sk_{0,1}$ to recursively invoke algorithm ComputeNext, i.e. compute

$$(sk_{w00,1}, sk_{w01,1}) = \mathsf{ComputeNext}(PK, sk_{w0,1}, w0), \text{ for all } 1 \leq |w0| \leq \kappa - 1.$$

Output: The algorithm outputs public key $PK = \left(1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\widetilde{\mathbb{B}}^{(k)}\}_{k=0,1,2}, \widehat{\mathbb{B}}^{*(1)}, \widehat{\mathbb{B}}^{*(2)}, \boldsymbol{b}^{*(0)}_4\right)$ and root secret key $SK_{0,1} = (sk_{0^\kappa,1}, \{sk_{1,1}, sk_{(01),1}, \ldots, sk_{(0^{\kappa-1}1),1}\})$.

Algorithm ComputeNext$(PK, sk_{w,l}, w)$: This is a helper method and is called by the Root Setup and Update algorithms. It takes a public key PK, a secret key $sk_{w,l}$, a node w, and outputs keys $sk_{w0,l}$, $sk_{w1,l}$ for time nodes $w0$ and $w1$ of predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. The algorithm proceeds as follows.

Parse w as $w_1, \ldots, w_r$, where $|w| = r$.

Parse $sk_{w,l}$ as $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},r+1}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},(2r+1)}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},L})$.

Computing $sk_{w0,l}$:

Pick $\psi, \psi', \epsilon^{(0)}_{\mathsf{dec}}, \epsilon^{(1)}_{\mathsf{dec},t}, \epsilon^{(1)}_{\mathsf{ran},j,t} (j = 1, \ldots, l+1), \epsilon^{(1)}_{\mathsf{del},j,t} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, l+1$.

Pick $\epsilon^{(2)}_{\mathsf{dec},t}, \sigma_{\mathsf{dec}}, \epsilon^{(2)}_{\mathsf{ran},j,t} (j = 1, \ldots, r+2), \sigma_{\mathsf{ran},j} (j = 1, \ldots, r+2), \epsilon^{(2)}_{\mathsf{del},j,t} (j = 1, \ldots, L), \sigma_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, r+1$.

Pick $\boldsymbol{r}^{(1)}_{\mathsf{dec}}, \boldsymbol{r}^{(1)}_{\mathsf{ran},j} (j = 1, \ldots, l+1), \boldsymbol{r}^{(1)}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n} \rangle$, $\boldsymbol{r}^{(2)}_{\mathsf{dec}}, \boldsymbol{r}^{(2)}_{\mathsf{ran},j} (j = 1, \ldots, r+2), \boldsymbol{r}^{(2)}_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1}, \ldots, \boldsymbol{b}^{*(2)}_{3L} \rangle$.

Compute

$$\begin{aligned}
\boldsymbol{k}^{(0)}_{w0,l,\mathsf{dec}} &= \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \epsilon^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_4, \\
\boldsymbol{k}^{(1)}_{w0,l,\mathsf{dec}} &= \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(1)}_{\mathsf{dec}}, \\
\boldsymbol{k}^{(2)}_{w0,l,\mathsf{dec}} &= \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{dec}} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \boldsymbol{r}^{(2)}_{\mathsf{dec}}, \\
\boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},j} &= \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(1)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, l+1, \\
\boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},j} &= \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{ran},j} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \boldsymbol{r}^{(2)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, r+2, \\
\boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},j} &= \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \psi \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(1)}_{\mathsf{del},j}, \text{ for } j = \mu_l + 1, \ldots, n, \\
\boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},j} &= \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{del},j} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \psi' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(2)}_{\mathsf{del},j}, \text{ for } j = 2(r+1)+1, \ldots, L.
\end{aligned}$$

Let $sk_{w0,l} = (\boldsymbol{k}^{(0)}_{w0,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},r+2},$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},\mu_l+1},\ldots,\boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},n},\boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},(2(r+1)+1)},\ldots,\boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},L}).$$

Computing $sk_{w1,l}$:

Pick $\tau,\tau',\varepsilon^{(0)}_{\mathsf{dec}},\varepsilon^{(1)}_{\mathsf{dec},t},\varepsilon^{(1)}_{\mathsf{ran},j,t}(j=1,\ldots,l+1),\varepsilon^{(1)}_{\mathsf{del},j,t}(j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t=1,\ldots,l+1$.

Pick $\varepsilon^{(2)}_{\mathsf{dec},t},\varsigma_{\mathsf{dec}},\varepsilon^{(2)}_{\mathsf{ran},j,t}(j=1,\ldots,r+2),\varsigma_{\mathsf{ran},j}(j=1,\ldots,r+2),\varepsilon^{(2)}_{\mathsf{del},j,t}(j=1,\ldots,L),\varsigma_{\mathsf{del},j}(j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t=1,\ldots,r+1$.

Pick $\boldsymbol{t}^{(1)}_{\mathsf{dec}},\boldsymbol{t}^{(1)}_{\mathsf{ran},j}(j=1,\ldots,l+1),\boldsymbol{t}^{(1)}_{\mathsf{del},j}(j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1},\ldots,\boldsymbol{b}^{*(1)}_{3n}\rangle$, $\boldsymbol{t}^{(2)}_{\mathsf{dec}},\boldsymbol{t}^{(2)}_{\mathsf{ran},j}(j=1,\ldots,r+2),\boldsymbol{t}^{(2)}_{\mathsf{del},j}(j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1},\ldots,\boldsymbol{b}^{*(2)}_{3L}\rangle$.

Compute

$$\begin{aligned}
\boldsymbol{k}^{(0)}_{w1,l,\mathsf{dec}} &= \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \varepsilon^{(0)}_{\mathsf{dec}}\boldsymbol{b}^{*(0)}_4,\\
\boldsymbol{k}^{(1)}_{w1,l,\mathsf{dec}} &= \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1}\varepsilon^{(1)}_{\mathsf{dec},t}\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{t}^{(1)}_{\mathsf{dec}},\\
\boldsymbol{k}^{(2)}_{w1,l,\mathsf{dec}} &= \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1}\varepsilon^{(2)}_{\mathsf{dec},t}\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{dec}}\left(\sum_{i=2r+1}^{2r+2}\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i}\right) + \boldsymbol{t}^{(2)}_{\mathsf{dec}},\\
\boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},j} &= \sum_{t=1}^{l+1}\varepsilon^{(1)}_{\mathsf{ran},j,t}\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{t}^{(1)}_{\mathsf{ran},j},\ \text{for } j=1,\ldots,l+1,\\
\boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},j} &= \sum_{t=1}^{r+1}\varepsilon^{(2)}_{\mathsf{ran},j,t}\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{ran},j}\left(\sum_{i=2r+1}^{2r+2}\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i}\right) + \boldsymbol{t}^{(2)}_{\mathsf{ran},j},\ \text{for } j=1,\ldots,r+2,\\
\boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},j} &= \sum_{t=1}^{l+1}\varepsilon^{(1)}_{\mathsf{del},j,t}\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \tau\boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{t}^{(1)}_{\mathsf{del},j},\ \text{for } j=\mu_l+1,\ldots,n,\\
\boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},j} &= \sum_{t=1}^{r+1}\varepsilon^{(2)}_{\mathsf{del},j,t}\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{del},j}\left(\sum_{i=2r+1}^{2r+2}\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i}\right) + \tau'\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{t}^{(2)}_{\mathsf{del},j},\\
&\qquad\qquad \text{for } j=2(r+1)+1,\ldots,L.
\end{aligned}$$

Let $sk_{w1,l} = (\boldsymbol{k}^{(0)}_{w1,l,\mathsf{dec}},\boldsymbol{k}^{(1)}_{w1,l,\mathsf{dec}},\boldsymbol{k}^{(2)}_{w1,l,\mathsf{dec}},\boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},1},\ldots,\boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},l+1},\boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},1},\ldots,\boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},r+2},$
$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},\mu_l+1},\ldots,\boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},n},\boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},(2(r+1)+1)},\ldots,\boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},L})$.

Output: Output $(sk_{w0,l},sk_{w1,l})$.

Algorithm $\mathsf{Delegate}(SK_{i,l},i,\overrightarrow{x}_{l+1}=(x_{\mu_l+1},\ldots,x_{\mu_{l+1}}))$: The delegation algorithm proceeds as follows.

Parse i as $i_1,\ldots,i_\kappa$ where $\kappa=\log_2 N$. Parse $SK_{i,l}$ as $(sk_{i,l},\{sk_{i|_{k-1}1,l}\}_{i_k=0})$.

For each $sk_{w,l}$ in $SK_{i,l}$ compute $sk_{w,l+1}$ as follows:

– Parse w as $w_1,\ldots,w_r$, where $|w|=r$.

– Pick $\psi,\psi',\gamma^{(0)}_{\mathsf{dec}},\gamma^{(1)}_{\mathsf{dec},t},\gamma^{(1)}_{\mathsf{ran},j,t}(j=1,\ldots,l+2),\gamma^{(1)}_{\mathsf{del},j,t}(j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t=1,\ldots,l+1$.

– Pick $\gamma^{(2)}_{\mathsf{dec},t}, \sigma_{\mathsf{dec}}, \gamma^{(2)}_{\mathsf{ran},j,t} (j = 1, \ldots, r+1), \sigma_{\mathsf{ran},j} (j = 1, \ldots, l+2), \gamma^{(2)}_{\mathsf{del},j,t} (j = 1, \ldots, L), \sigma_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, r+1$.

– Pick $\boldsymbol{r}^{(1)}_{\mathsf{dec}}, \boldsymbol{r}^{(1)}_{\mathsf{ran},j} (j = 1, \ldots, l+2), \boldsymbol{r}^{(1)}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n} \rangle$, $\boldsymbol{r}^{(2)}_{\mathsf{dec}}, \boldsymbol{r}^{(2)}_{\mathsf{ran},j} (j = 1, \ldots, r+1), \boldsymbol{r}^{(2)}_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1}, \ldots, \boldsymbol{b}^{*(2)}_{3L} \rangle$.

– Compute

$$\boldsymbol{k}^{(0)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \gamma^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_4,$$

$$\boldsymbol{k}^{(1)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \gamma^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{dec}} \Big(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},i} \Big) + \boldsymbol{r}^{(1)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \gamma^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(2)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(1)}_{w,l+1,\mathsf{ran},j} = \sum_{t=1}^{l+1} \gamma^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{ran},j} \Big(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},i} \Big) + \boldsymbol{r}^{(1)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, l+2,$$

$$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{ran},j} = \sum_{t=1}^{r+1} \gamma^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(2)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, r+1,$$

$$\boldsymbol{k}^{(1)}_{w,l+1,\mathsf{del},j} = \sum_{t=1}^{l+1} \gamma^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{del},j} \Big(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},i} \Big) + \psi \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(1)}_{\mathsf{del},j},$$
$$\text{for } j = \mu_{l+1} + 1, \ldots, n,$$

$$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{del},j} = \sum_{t=1}^{r+1} \gamma^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \psi' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(2)}_{\mathsf{del},j}, \text{ for } j = 2r+1, \ldots, L.$$

Let $sk_{w,l+1} = (\boldsymbol{k}^{(0)}_{w,l+1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{ran},l+2}, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{ran},1}, \ldots,$
$\boldsymbol{k}^{(2)}_{w,l+1,\mathsf{ran},r+1}, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{del},\mu_{l+1}+1}, \ldots, \boldsymbol{k}^{(1)}_{w,l+1,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{del},2r+1}, \ldots, \boldsymbol{k}^{(2)}_{w,l+1,\mathsf{del},L})$.

Output $SK_{i,l+1} = (sk_{i,l+1}, \{sk_{i|_{k-1}1,l+1}\}_{i_k=0})$ and erase all other information.

Algorithm $\mathsf{Update}(SK_{i,l}, i)$: This algorithm follows the concept from [11, 22] to compute a private key for the next time period $i+1$. Parse i as $i_1, \ldots, i_\kappa$ where $|i| = \kappa$. Parse $SK_{i,l}$ as $(sk_{i,l}, \{sk_{i|_{k-1}1,l}\}_{i_k=0})$. Erase $sk_{i,l}$. If $i_\kappa = 0$, simply output the remaining keys as the key $SK_{(i+1),l}$ for the next period. Otherwise, let $\tilde{k}$ be the largest value such that $i_{\tilde{k}} = 0$. Let $i' = i|_{\tilde{k}-1}1$. Using $sk_{i',l}$, which is part of $SK_{i,l}$, recursively apply algorithm $\mathsf{ComputeNext}$ to generate keys $sk_{(i'0^d1),l}$ for $0 \leq d \leq l - \tilde{k} - 1$ and $sk_{(i'0^{d-\tilde{k}},l)}$. Erase $sk_{i',l}$ and output the remaining keys as $SK_{(i+1),l}$.

Remark 5. Note that the key $sk_{(i'0^{d-\tilde{k}},l)}$ will be used for decryption in the next time period $i+1$, whereas other generated secret keys will be used to compute private key of the next period.

Algorithm $\mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = ((y_1, \ldots, y_{\mu_1}), \ldots, (y_{\mu_{h-1}+1}, \ldots, y_{\mu_h})), i, M \in \mathbb{G}_T)$: The encryption algorithms proceeds as follows.

Parse i as $i_1, \ldots, i_\kappa$.
Pick $(\overrightarrow{y}_{h+1}, \ldots, \overrightarrow{y}_d) \xleftarrow{\mathsf{U}} \mathbb{F}_q^{\mu_{h+1}-\mu_h} \times \ldots \times \mathbb{F}_q^{n-\mu_{d-1}}$, $\delta, \zeta, \varphi, \varphi^{(1)}, \varphi^{(2)} \xleftarrow{\mathsf{U}} \mathbb{F}_q$.

Compute the ciphertext components

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, 0, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_d), 0^{2n}, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1, -i_1), \ldots, (1, -i_\kappa)), 0^{2L}, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^{\zeta} M.
\end{aligned}$$

Output ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$.

Algorithm Decrypt$(C, SK_{i,l})$: Parse C as $(\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$. Parse $SK_{i,l}$ as $(sk_{i,l}, \{sk_{i|_{k-1}1,l}\}_{i_k=0})$. Use $sk_{i,l}$ to decrypt and output

$$M = \frac{\boldsymbol{c}^{(M)}}{e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{(0)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}^{(1)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(2)}, \boldsymbol{k}^{(2)}_{i,l,\mathsf{dec}})}.$$

4.3 Correctness

To see why the scheme is correct, let C and $SK_{i,l}$ be as above. If $\overrightarrow{x}_i \cdot \overrightarrow{y}_i = 0$ for $1 \leq i \leq l$, and C and $SK_{i,l}$ are encoded with the same time period i then M can be recovered as specified in the decryption algorithm due to the following equality

$$e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{(0)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}^{(1)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(2)}, \boldsymbol{k}^{(2)}_{i,l,\mathsf{dec}}) = g_T^{-\alpha_{\mathsf{dec}}\delta+\zeta} g_T^{\alpha^{(1)}_{\mathsf{dec}}\delta} g_T^{\alpha^{(2)}_{\mathsf{dec}}\delta} = g_T^{-\alpha\delta+\zeta} g_T^{\alpha\delta} = g_T^{\zeta}.$$

In the scheme, the size for the secret key $SK_{i,l}$ is $(3n^2 + 12\kappa^2 + 3nl + 7n - 3n\mu_l - \mu_l + 14\kappa - 6r\kappa + r + 9)|\mathbb{G}|$, and the size of the ciphertext is $(3n + 6\kappa + 8)|\mathbb{G}| + |\mathbb{G}_T|$, where n denotes the size of predicate vectors, κ is depth of the hierarchy, l represents the level of hierarchical predicate, μ_l is the size of level l-predicate and r is the level of time node, $|\mathbb{G}|$ is the size of group element in $\mathbb{G}$, and $|\mathbb{G}_T|$ is the size of group element in $\mathbb{G}_T$.

4.4 Security Analysis

The security of our FS-HPE scheme is established through the following theorem.

Theorem 1. *Our FS-HPE scheme is adaptively attribute-hiding against chosen plaintext attacks under the DLIN assumption. For any adversary $\mathcal{A}$, there exists a probabilistic polynomial time machine $\mathcal{D}$ such that for any security parameter λ,*

$$\mathsf{Adv}^{\mathsf{FS\text{-}HPE}}_{\mathcal{A}}(\lambda) \leq (2\nu(\kappa+1)(n+L+1)+1)\mathsf{Adv}^{\mathrm{DLIN}}_{\mathcal{D}}(\lambda) + \psi,$$

where ν is the maximum number of $\mathcal{A}$'s key queries, κ is the depth of the time tree, and $\psi = (20\nu(\kappa+1)(n+L+1)+9)/q$.

The proof of Theorem 1 is available in Appendix B.

5 Conclusion

In this paper, we introduced the accepted notion of Forward Security to a powerful setting of Hierarchical Predicate Encryption. The resulting FS-HPE scheme offers time-independent delegation of predicates, autonomous update for users' private keys, and its encryption process doesn't require knowledge of time periods when particular predicates joined the predicate hierarchy. The scheme is forward-secure and adaptively attribute-hiding under chosen plaintext attacks under the DLIN assumption in the standard model.

Acknowledgements

This research work is part of the bilateral research project between Germany and Australia, funded jointly by the German Academic Exchange Service (DAAD) through grant Nr. 53361649 and by Australia's Department of Innovation, Industry, Science and Research (DIISR). Mark Manulis was also supported by the German Research Foundation (DFG) through grant MA 4096. He wishes further to acknowledge support from the Center of Advanced Security Research Darmstadt (CASED) and the European Center for Security and Privacy by Design (EC SPRIDE).

References

1. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. *J. Cryptology*, 21(3):350–391, 2008.
2. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer-Verlag, 2001.
3. Mihir Bellare and Sara K. Miner. A forward-secure digital signature scheme. In *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 431–448. Springer-Verlag, 1999.
4. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.
5. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology – EUROCRYPT 2005*, Lecture Notes in Computer Science, pages 440–456. Springer-Verlag, 2005.
6. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.
7. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 506–522. Springer-Verlag, 2004.
8. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil Pairing. In *CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer-Verlag, 2001.
9. Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 535–554. Springer-Verlag, 2007.
10. Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology - CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 290–307. Springer-Verlag, 2006.
11. Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Advances in Cryptology – EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 255–271. Springer-Verlag, 2003.
12. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222. Springer-Verlag, 2004.
13. Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Fully secure anonymous hibe and secret-key anonymous ibe with short ciphertexts. In *Pairing*, volume 6487 of *Lecture Notes in Computer Science*, pages 347–366. Springer-Verlag, 2010.
14. Melissa Chase. Multi-authority attribute based encryption. In *TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 515–534. Springer-Verlag, 2007.
15. Whitfield Diffie, Paul C. Van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2:107–125, 1992.
16. Léo Ducas. Anonymity from asymmetry: New constructions for anonymous hibe. In *CT-RSA*, volume 5985 of *Lecture Notes in Computer Science*, pages 148–164. Springer-Verlag, 2010.
17. Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566. Springer-Verlag, 2002.
18. Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS 2006*, pages 89–98. ACM, 2006.
19. Christoph G. Günther. An identity-based key-exchange protocol. In *In Advances in Cryptology –Eurocrypt'89*, volume 434 of *Lecture Notes in Computer Science*, pages 29–37. Springer-Verlag, 1989.
20. Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481. Springer-Verlag, 2002.
21. Gene Itkis and Leonid Reyzin. Forward-Secure Signatures with Optimal Signing and Verifying. In *CRYPTO 2001*, volume 2139 of *LNCS*, pages 332–354. Springer, 2001.

22. Jonathan Katz. A forward-secure public-key encryption scheme. Cryptology ePrint Archive, Report 2002/060, 2002. http://eprint.iacr.org/.
23. Jonathan Katz. Binary Tree Encryption: Constructions and Applications. In *Information Security and Cryptology (ICISC 2003)*, volume 2971 of *LNCS*, pages 1–11. Springer, 2003.
24. Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 2008.
25. Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer-Verlag, 2010.
26. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC 2010*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479. Springer-Verlag, 2010.
27. Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT 2009*, volume 5912 of *Lecture Notes in Computer Science*, pages 214–231. Springer-Verlag, 2009.
28. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer-Verlag, 2010.
29. Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. Cryptology ePrint Archive, Report 2010/563, 2010. http://eprint.iacr.org/.
30. Tatsuaki Okamoto and Katsuyuki Takashima. Achieving short ciphertexts or short secret-keys for adaptively secure general inner-product encryption. In *CANS*, volume 7092 of *Lecture Notes in Computer Science*, pages 138–159. Springer-Verlag, 2011.
31. Tatsuaki Okamoto and Katsuyuki Takashima. Adaptively attribute-hiding (hierarchical) inner product encryption. In *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 591–608. Springer-Verlag, 2012.
32. Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 457–473. Springer-Verlag, 2005.
33. Jae Hong Seo and Jung Hee Cheon. Fully secure anonymous hierarchical identity-based encryption with constant size ciphertexts. Cryptology ePrint Archive, Report 2011/021, 2011. http://eprint.iacr.org/.
34. Jae Hong Seo, Tetsutaro Kobayashi, Miyako Ohkubo, and Koutarou Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In *Public Key Cryptography - PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 215–234. Springer-Verlag, 2009.
35. Adi Shamir. Identity based cryptosystems and signature schemes. In *CRYPTO'84*, volume 0196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1984.
36. Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC 2009*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer-Verlag, 2009.
37. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *ICALP 2008 (2)*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578. Springer-Verlag, 2008.
38. Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer-Verlag, 2009.
39. Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the 11th ACM Conference on Computer and Communications Security – CCS 2004*, pages 354–363. ACM, 2004.

A Hierarchical Predicate Encryption

We include the syntax for Hierarchical Predicate Encryption (HPE) [25], followed by the HPE scheme.

A.1 Syntax

Definition 5. *Let $\overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)$ s.t. $\mu_0 = 0 < \mu_1 < \mu_2 < \ldots < \mu_d = n$ be a format of hierarchy of depth d attribute spaces. A* Hierarchical Predicate Encryption (HPE) *comprises of five algorithms* (Setup, Genkey, Encrypt, Decrypt, Delegate).

- $\mathsf{Setup}(1^\lambda, \overrightarrow{\mu})$ *This algorithm takes as input a security parameter 1^λ and the format of hierarchy $\overrightarrow{\mu}$. It outputs a public key PK and a master secret key SK.*
- $\mathsf{Genkey}(PK, SK, (\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l))$ *This algorithm takes as input a public key PK, a master secret key SK, and predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. It outputs a secret key $sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$.*
- $\mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h), M)$ *This algorithm takes as input a public key PK, hierarchical attribute vectors $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$, and a message M in some associated message space. It outputs a ciphertext C.*
- $\mathsf{Decrypt}(C, sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)})$ *This algorithm takes as input a ciphertext C and a secret key $sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$ for attribute vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. It outputs either a message M or the distinguished symbol $\perp$.*
- $\mathsf{Delegate}(PK, sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}, \overrightarrow{x}_{l+1})$ *This algorithm takes as input the public key pk, l-th level secret key $sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$, and $(l+1)$-th level predicate vector $\overrightarrow{x}_{l+1}$. It outputs $(l+1)$-th level secret key $sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_{l+1})}$.*

Correctness. *We require the following correctness property: for all correctly generated PK and $sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}$ associated with predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$, generate $C \xleftarrow{R} \mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h), M)$ and $M' = \mathsf{Decrypt}(C, sk_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)})$. If $f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h) = 1$, then $M = M'$. Otherwise, $M \neq M'$ with all but negligible probability.*

A.2 Underlying HPE Scheme

Our forward secure HPE scheme is based on the following variant of the HPE scheme from [25]. Informally, l-th level secret key $sk_l = (\boldsymbol{k}^{(0)}_{l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{l,\mathsf{ran},l+1}, \boldsymbol{k}^{(1)}_{l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{l,\mathsf{del},n})$ consists of three components. The decryption component $(\boldsymbol{k}^{(0)}_{l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{l,\mathsf{dec}})$, the randomness component $(\boldsymbol{k}^{(1)}_{l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{l,\mathsf{ran},l+1})$ and delegation component $(\boldsymbol{k}^{(1)}_{l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{l,\mathsf{del},n})$. The decryption component is used to decrypt the ciphertext. The randomness component is used to re-randomize the coefficients of the delegated key, and the delegation component is essential to delegate the decryption rights to the $l+1$-th level child key. We refer to [25] for the key idea of constructing the HPE.

$\mathsf{Setup}\big(1^\lambda, \overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)\big)$: The initialization algorithm proceeds as follows.

Let $\overrightarrow{n} = (1; n)$, $(\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}) \xleftarrow{R} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n})$.

$\widetilde{\mathbb{B}}^{(0)} = (\boldsymbol{b}^{(0)}_1, \boldsymbol{b}^{(0)}_3, \boldsymbol{b}^{(0)}_5)$, $\widetilde{\mathbb{B}}^{(1)} = (\boldsymbol{b}^{(1)}_1, \ldots, \boldsymbol{b}^{(1)}_n, \boldsymbol{b}^{(1)}_{3n+1})$,

$\widetilde{\mathbb{B}}^{*(0)} = (\boldsymbol{b}^{*(0)}_1, \boldsymbol{b}^{*(0)}_3)$, $\widetilde{\mathbb{B}}^{*(1)} = (\boldsymbol{b}^{*(1)}_1, \ldots, \boldsymbol{b}^{*(1)}_n)$, $\widehat{\mathbb{B}}^{*(1)} = (\boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n})$,

Output public key $PK = \big(1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\widetilde{\mathbb{B}}^{(k)}\}_{k=0,1}, \widehat{\mathbb{B}}^{*(1)}, \boldsymbol{b}^{*(0)}_4\big)$ and secret key $SK = \{\widetilde{\mathbb{B}}^{*(k)}\}_{k=0,1}$.

$\mathsf{GenKey}(PK, MSK, (\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l) = ((x_1, \ldots, x_{\mu_1}), \ldots, (x_{\mu_{l-1}+1}, \ldots, x_{\mu_l})))$: The key generation algorithm contains the following steps.

Pick $\psi, \alpha_{\mathsf{dec}}, \eta^{(0)}_{\mathsf{dec}}, \beta^{(1)}_{\mathsf{dec},t}, \beta^{(1)}_{\mathsf{ran},j,t} (j = 1, \ldots, l+1), \beta^{(1)}_{\mathsf{del},j,t} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\overrightarrow{\eta}^{(1)}_{\mathsf{dec}}$, $\overrightarrow{\eta}^{(1)}_{\mathsf{ran},j} (j = 1, \ldots, l+1)$, $\overrightarrow{\eta}^{(1)}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}^n_q$, for $t = 1, \ldots, l$.

Compute

$$\begin{aligned}
&\boldsymbol{k}_{l,\mathsf{dec}}^{(0)} = (-\alpha_{\mathsf{dec}}, 0, 1, \eta_{\mathsf{dec}}^{(0)}, 0)_{\mathbb{B}^{*(0)}},\\
&\boldsymbol{k}_{l,\mathsf{dec}}^{(1)} = (\alpha_{\mathsf{dec}} \overrightarrow{e}_1^{(1)} + \beta_{\mathsf{dec},1}^{(1)} \overrightarrow{x}_1, \beta_{\mathsf{dec},2}^{(1)} \overrightarrow{x}_2, \ldots, \beta_{\mathsf{dec},l}^{(1)} \overrightarrow{x}_l, 0^{2n-\mu_l}, \overrightarrow{\eta}_{\mathsf{dec}}^{(1)}, 0)_{\mathbb{B}^{*(1)}},\\
&\boldsymbol{k}_{l,\mathsf{ran},j}^{(1)} = (\beta_{\mathsf{ran},j,1}^{(1)} \overrightarrow{x}_1, \beta_{\mathsf{ran},j,2}^{(1)} \overrightarrow{x}_2, \ldots, \beta_{\mathsf{ran},j,l}^{(1)} \overrightarrow{x}_l, 0^{2n-\mu_l}, \overrightarrow{\eta}_{\mathsf{ran},j}^{(1)}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j = 1, \ldots, l+1,\\
&\boldsymbol{k}_{l,\mathsf{del},j}^{(1)} = (\beta_{\mathsf{del},j,1}^{(1)} \overrightarrow{x}_1, \beta_{\mathsf{del},j,2}^{(1)} \overrightarrow{x}_2, \ldots, \beta_{\mathsf{del},j,l}^{(1)} \overrightarrow{x}_l, 0^{j-\mu_l-1}, \psi, 0^{2n-j}, \overrightarrow{\eta}_{\mathsf{del},j}^{(1)}, 0)_{\mathbb{B}^{*(1)}}, \text{ for } j = \mu_l+1, \ldots, n,
\end{aligned}$$

Output $sk_l = (\boldsymbol{k}_{l,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{l,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{l,\mathsf{ran},1}^{(1)}, \ldots, \boldsymbol{k}_{l,\mathsf{ran},l+1}^{(1)}, \boldsymbol{k}_{l,\mathsf{del},\mu_l+1}^{(1)}, \ldots, \boldsymbol{k}_{l,\mathsf{del},n}^{(1)})$.

$\mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_l) = ((y_1, \ldots, y_{\mu_1}), \ldots, (y_{\mu_{l-1}+1}, \ldots, y_{\mu_l})), M \in \mathbb{G}_T)$: The encryption algorithm proceeds as follows.

Pick $(\overrightarrow{y}_{l+1}, \ldots, \overrightarrow{y}_d) \xleftarrow{\mathsf{U}} \mathbb{F}_q^{\mu_{l+1}-\mu_l} \times \ldots \times \mathbb{F}_q^{n-\mu_{d-1}}, \delta, \zeta, \varphi, \varphi^{(1)}, \delta_1, \ldots, \delta_d \xleftarrow{\mathsf{U}} \mathbb{F}_q$,

$$\begin{aligned}
&\boldsymbol{c}^{(0)} = (\delta, 0, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}},\\
&\boldsymbol{c}^{(1)} = (\delta_1 \overrightarrow{y}_1, \ldots, \delta_d \overrightarrow{y}_d, 0^{2n}, \varphi^{(1)})_{\mathbb{B}^{(1)}},\\
&\boldsymbol{c}^{(M)} = g_T^{\zeta} M.
\end{aligned}$$

Output ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(M)})$.

$\mathsf{Decrypt}(C, sk_l)$: The decryption algorithm on input a ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(M)})$ and a secret key $sk_l = (\boldsymbol{k}_{l,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{l,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{l,\mathsf{ran}}^{(0)}, \boldsymbol{k}_{l,\mathsf{ran},j}^{(1)}, \ldots, \boldsymbol{k}_{l,\mathsf{ran},l+1}^{(1)}, \boldsymbol{k}_{l,\mathsf{del},\mu+1}^{(1)}, \ldots, \boldsymbol{k}_{l,\mathsf{del},n}^{(1)})$ outputs

$$M = \frac{\boldsymbol{c}^{(M)}}{e(\boldsymbol{c}^{(0)}, \boldsymbol{k}_{l,\mathsf{dec}}^{(0)}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}_{l,\mathsf{dec}}^{(1)})}.$$

$\mathsf{Delegate}(PK, sk_l, \overrightarrow{x}_{l+1} = (x_{\mu_l+1}, \ldots, x_{\mu_{l+1}}))$: The delegation algorithm executes the following steps.

Pick $\psi', \gamma_{\mathsf{dec}}, \sigma_{\mathsf{dec}}, \gamma_{\mathsf{dec},t}, \gamma_{\mathsf{ran},j,t}, \sigma_{\mathsf{ran},j} (j = 1, \ldots, l+2), \gamma_{\mathsf{del},j,t}, \sigma_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $t = 1, \ldots, l+1$.
Pick $\boldsymbol{r}_{\mathsf{dec}}, \boldsymbol{r}_{\mathsf{ran},j} (j = 1, \ldots, l+2), \boldsymbol{r}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}_{2n+1}^{*(1)}, \ldots, \boldsymbol{b}_{3n}^{*(1)} \rangle$.

Compute

$$\begin{aligned}
&\boldsymbol{k}_{l+1,\mathsf{dec}}^{(0)} = \boldsymbol{k}_{l,\mathsf{dec}}^{(0)} + \gamma_{\mathsf{dec}} \boldsymbol{b}_4^{*(0)},\\
&\boldsymbol{k}_{l+1,\mathsf{dec}}^{(1)} = \boldsymbol{k}_{l,\mathsf{dec}}^{(1)} + \textstyle\sum_{t=1}^{l+1} \gamma_{\mathsf{dec},t} \boldsymbol{k}_{l,\mathsf{ran},t}^{(1)} + \sigma_{\mathsf{dec}} \left(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}_{l,\mathsf{del},i}^{(1)} \right) + \boldsymbol{r}_{\mathsf{dec}},\\
&\boldsymbol{k}_{l+1,\mathsf{ran},j}^{(1)} = \textstyle\sum_{t=1}^{l+1} \gamma_{\mathsf{ran},j,t} \boldsymbol{k}_{l,\mathsf{ran},t}^{(1)} + \sigma_{\mathsf{ran},j} \left(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}_{l,\mathsf{del},i}^{(1)} \right) + \boldsymbol{r}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, l+2,\\
&\boldsymbol{k}_{l+1,\mathsf{del},j}^{(1)} = \textstyle\sum_{t=1}^{l+1} \gamma_{\mathsf{del},j,t} \boldsymbol{k}_{l,\mathsf{ran},t}^{(1)} + \sigma_{\mathsf{del},j} \left(\sum_{i=\mu_l+1}^{\mu_{l+1}} x_i \boldsymbol{k}_{l,\mathsf{del},i}^{(1)} \right) + \psi' \boldsymbol{k}_{l,\mathsf{del},j}^{(1)} + \boldsymbol{r}_{\mathsf{del},j}, \text{ for } j = \mu_{l+1}+1, \ldots, n,
\end{aligned}$$

Output $\overrightarrow{\boldsymbol{k}}_{l+1} = (\boldsymbol{k}_{l+1,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{l+1,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{l+1,\mathsf{ran},1}^{(1)}, \ldots, \boldsymbol{k}_{l+1,\mathsf{ran},l+2}^{(1)}, \boldsymbol{k}_{l+1,\mathsf{del},\mu_{l+1}+1}^{(1)}, \ldots, \boldsymbol{k}_{l+1,\mathsf{del},n}^{(1)})$.

B Proof of Theorem 1

Outline of the Proof of Theorem 1: In the proof, we have semi-functional ciphertexts and keys in addition to normal ciphertexts and keys. Semi-functional keys can decrypt all normal ciphertexts, but decryption will fail if one attempts to decrypt semi-functional ciphertexts with semi-functional keys. Similarly, semi-functional ciphertexts can be decrypted only by normal keys. A normal secret key $SK_{i,l} = (sk_{i,l}, \{sk_{i|_{k-1},1,l}\}_{i_k=0})$ associated with time period i and hierarchical predicate $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$, a normal ciphertext C with hierarchical predicate $(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h)$ and time

period i are shown in the scheme. A semi-functional secret key and a semi-functional ciphertext are expressed in Eqs. (9)-(11) and Eq.(12) respectively. We also introduce nominal semi-functional form which is analogue to that in [26]. A nominal semi-functional secret key and a nominal semi-functional ciphertext are expressed in in Eqs. (5)-(7) and Eq.(8) respectively. Normal ciphertexts and keys are used in the real system, while their nominal semi-functional or semi-functional counterparts are used in a sequence of security games only.

To prove the theorem, we analyze a sequence of games from Game 0 (original game) to Game 3. The challenge ciphertext is changed to a semi-functional one in Game 1. When at most ν delegation queries are issued and let κ be the depth of the time tree, there are $2\nu(\kappa+1)(n+L+1)$ game changes from Game 1 (Game 2-$(0,0,0)$), Game 2-$(0,0,0')$, Game 2-$(0,0,1)$, Game 2-$(0,0,1')$, Game 2-$(0,0,2)$, Game 2-$(0,0,2')$ through Game 2-$(\nu-1,\kappa,(n+L)')$ and Game 2-$(\nu-1,\kappa,n+L+1)$ (Game 2-$(\nu,0,0)$). In Game 2-(m,k,j), the first $m\times k\times j$ keys are semi-functional and the rest of keys are normal, and challenge ciphertext is semi-functional. In Game 2-(m,k,j'), the first $m\times k\times j$ keys are semi-functional and the $(m\times k\times j+1)$-th key is nominal semi-functional while the remaining keys are normal, and challenge ciphertext is nominal semi-functional. In Game 3, all keys and challenge ciphertext are semi-functional, where the adversary has 0 advantage.

The advantage difference between Games 0 and 1 is equivalent to the advantage of Problem 1. To prove that, we construct a simulator that uses a Problem 1 instance as an input, and interpolates between Game 0 and Game 1. We show that the distribution of the secret keys and the challenge ciphertext answered by the simulator is identical to those of Game 0 provided $\beta=0$ and those of Game 1 provided $\beta=1$. The hardness of Problem 1 is also based on the DLIN assumption.

The advantage difference between Game 2-(m,k,j) and Game 2-(m,k,j'), is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption). Here, we introduce special forms of nominal semi-functional keys and ciphertext. They are same as their counterparts in semi-functional forms except that $\epsilon w=\gamma=\gamma^{(1)}+\gamma^{(2)}$ and $\epsilon\xleftarrow{\mathsf{U}}\mathbb{F}_q$. Semi-functional keys and ciphertext are simulated using Problem 2 instance when $\beta=1$. Due to their algebra structures, semi-functional keys can always decrypt Semi-functional ciphertext when $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}_1,\ldots,\overrightarrow{y}_h)=1$ and time periods match. Therefore, it is hard for the simulator to test for the semi-functional key by creating the semi-functional ciphertext by itself. On the other hand, γ is independently distributed from the other variables when either the predicate does not hold or time for queried key is greater than the challenge time. That is, the joint distribution of nominal semi-functional key and ciphertext is equivalent to that of semi-functional key and ciphertext when either condition holds. Hence, both of them appear identical from the adversary's view, since from the security definition the adversary's queries should satisfy at least one of the conditions (predicate does not holds and time for queried key is greater than the challenge time).

With the similar argument, we show that the advantage difference between Game 2-(m,k,j') and Game 2-$(m,k,j+1)$ is equivalent to the advantage of Problem 2 (i.e., advantage of the DLIN assumption).

In the final step, we show that Game 2-$(\nu,0,0)$ can be conceptually changed to Game 3.

Definition 6 (Problem 1). *Problem 1 is to find bit β given* $(\mathsf{param}_{\overrightarrow{n}},\{\mathbb{B}^{(k)},\widehat{\mathbb{B}}^{*(k)}\}_{k=0,1,2},\boldsymbol{t}_{\beta}^{(0)},\{\boldsymbol{t}_{\beta,1}^{(k)}\}_{k=1,2},\{\boldsymbol{t}_{i}^{(k)}\}_{i=2,\ldots,n_k;k=1,2})\xleftarrow{\mathsf{R}}\mathcal{G}_{\beta}^{\mathrm{P1}}(1^{\lambda},\overrightarrow{n}=(2;n_1,n_2))$ *for $\beta\xleftarrow{\mathsf{U}}\{0,1\}$ with probability non-negligibly better than by a random guess, where*

$$
\begin{aligned}
&\mathcal{G}_{\beta}^{\mathrm{P1}}(1^{\lambda},\overrightarrow{n}=(2;n_1,n_2)):\ (\mathsf{param}_{\overrightarrow{n}},\mathbb{B}^{(0)},\mathbb{B}^{*(0)},\mathbb{B}^{(1)},\mathbb{B}^{*(1)},\mathbb{B}^{(2)},\mathbb{B}^{*(2)})\xleftarrow{\mathsf{R}}\mathcal{G}_{\mathsf{ob}}(1^{\lambda},\overrightarrow{n}),\\
&\widehat{\mathbb{B}}^{*(0)}=(\boldsymbol{b}_1^{*(0)},\boldsymbol{b}_3^{*(0)},\boldsymbol{b}_4^{*(0)},\boldsymbol{b}_5^{*(0)}),\\
&\widehat{\mathbb{B}}^{*(1)}=(\boldsymbol{b}_1^{*(1)},\ldots,\boldsymbol{b}_{n_1}^{*(1)},\boldsymbol{b}_{2n_1+1}^{*(1)},\ldots,\boldsymbol{b}_{3n_1+1}^{*(1)}),\\
&\widehat{\mathbb{B}}^{*(2)}=(\boldsymbol{b}_1^{*(2)},\ldots,\boldsymbol{b}_{n_2}^{*(2)},\boldsymbol{b}_{2n_2+1}^{*(2)},\ldots,\boldsymbol{b}_{3n_2+1}^{*(2)}),\\
&\delta,u,\rho\xleftarrow{\mathsf{U}}\mathbb{F}_q,\ \ \boldsymbol{t}_0^{(0)}=(\delta,0,0,0,\rho)_{\mathbb{B}^{(0)}},\ \ \boldsymbol{t}_1^{(0)}=(\delta,u,0,0,\rho)_{\mathbb{B}^{(0)}},\\
&\text{For } k=1,2:\\
&\quad\rho^{(k)}\xleftarrow{\mathsf{U}}\mathbb{F}_q,\ \overrightarrow{u}^{(k)}\xleftarrow{\mathsf{U}}\mathbb{F}_q^{n_k},\\
&\quad\boldsymbol{t}_{0,1}^{(k)}=(\overbrace{\delta\overrightarrow{e}_1^{(k)}}^{n_k},\overbrace{0^{n_k}}^{n_k},\overbrace{0^{n_k}}^{n_k},\overbrace{\rho^{(k)}}^{1})_{\mathbb{B}^{(k)}},
\end{aligned}
$$

$$\boldsymbol{t}_{1,1}^{(k)} = (\overbrace{\delta\overrightarrow{e}_1^{(k)}}^{n_k}, \overbrace{\overrightarrow{u}^{(k)}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{\rho^{(k)}}^{1})_{\mathbb{B}^{(k)}},$$

For $i = 2, \ldots, n_k$: $\boldsymbol{t}_i^{(k)} = \delta \boldsymbol{b}_i^{(k)}$,

return $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(k)}, \widehat{\mathbb{B}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{t}_\beta^{(0)}, \{\boldsymbol{t}_{\beta,1}^{(k)}\}_{k=1,2}, \{\boldsymbol{t}_i^{(k)}\}_{i=2,\ldots,n_k;k=1,2})$.

The corresponding advantage of PPT algorithm $\mathcal{B}$ in solving Problem 1 is defined as follows:

$$\mathsf{Adv}_{\mathcal{B}}^{\mathrm{P1}}(\lambda) = \Big| \mathsf{Pr}\Big[\mathcal{B}(1^\lambda, \varpi) \to 1 \;\Big|\; \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathrm{P1}}(1^\lambda, \overrightarrow{n})\Big] \;-\; \mathsf{Pr}\Big[\mathcal{B}(1^\lambda, \varpi) \to 1 \;\Big|\; \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathrm{P1}}(1^\lambda, \overrightarrow{n})\Big] \Big|.$$

Lemma 1. *For any adversary $\mathcal{B}$, there exists a probabilistic machine $\mathcal{D}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{B}}^{\mathrm{P1}}(\lambda) \leq \mathsf{Adv}_{\mathcal{D}}^{\mathrm{DLIN}}(\lambda) + 8/q$.*

The proof for Lemma 1 can be found in section B.1.

Definition 7 (Problem 2). *Problem 2 is to find bit $\beta \in \{0,1\}$, given* $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{h}_\beta^{*(0)}, \boldsymbol{t}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{h}_{\beta,i}^{*(k)}, \boldsymbol{t}_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2}) \xleftarrow{R} \mathcal{G}_\beta^{\mathrm{P2}}(1^\lambda, \overrightarrow{n} = (2; n_1, n_2))$, *where*

$$\mathcal{G}_\beta^{\mathrm{P2}}(1^\lambda, \overrightarrow{n} = (2; n_1, n_2)) : (\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{*(2)}) \xleftarrow{R} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}),$$

$$\widehat{\mathbb{B}}^{(0)} = (\boldsymbol{b}_1^{(0)}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_4^{(0)}, \boldsymbol{b}_5^{(0)}),$$

$$\widehat{\mathbb{B}}^{(1)} = (\boldsymbol{b}_1^{(1)}, \ldots, \boldsymbol{b}_{n_1}^{(1)}, \boldsymbol{b}_{2n_1+1}^{(1)}, \ldots, \boldsymbol{b}_{3n_1+1}^{(1)}),$$

$$\widehat{\mathbb{B}}^{(2)} = (\boldsymbol{b}_1^{(2)}, \ldots, \boldsymbol{b}_{n_2}^{(2)}, \boldsymbol{b}_{2n_2+1}^{(2)}, \ldots, \boldsymbol{b}_{3n_2+1}^{(2)}),$$

$$\omega, \xi, \delta \xleftarrow{\mathsf{U}} \mathbb{F}_q, \;\; z, \pi \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times, \;\; u = z^{-1},$$

$$\boldsymbol{h}_0^{*(0)} = (\omega, 0, 0, \xi, 0)_{\mathbb{B}^{*(0)}}, \;\; \boldsymbol{h}_1^{*(0)} = (\omega, z, 0, \xi, 0)_{\mathbb{B}^{*(0)}}, \;\; \boldsymbol{t}^{(0)} = (\delta, \pi u, 0, 0, 0)_{\mathbb{B}^{(0)}},$$

For $k = 1, 2$:

For $i = 1, \ldots, n_k$ and $j = 1, \ldots, n_k$:

$$(u_{i,j}^{(k)}) \xleftarrow{\mathsf{U}} GL(\mathbb{F}_q, n_k), \; (z_{i,j}^{(k)}) = \big((u_{i,j}^{(k)})^{-1}\big)^T;$$

For $i = 1, \ldots, n_k$:

$$\overrightarrow{\omega}_i^{(k)} \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_k},$$

$$\boldsymbol{h}_{0,i}^{*(k)} = (\overbrace{\omega\overrightarrow{e}_i^{(k)}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{\overrightarrow{\omega}_i^{(k)}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}},$$

$$\boldsymbol{h}_{1,i}^{*(k)} = (\overbrace{\omega\overrightarrow{e}_i^{(k)}}^{n_k}, \overbrace{z_{i,1}^{(k)}, \ldots, z_{i,n_k}^{(k)}}^{n_k}, \overbrace{\overrightarrow{\omega}_i^{(k)}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}},$$

$$\boldsymbol{t}_i^{(k)} = (\overbrace{\delta\overrightarrow{e}_i^{(k)}}^{n_k}, \overbrace{\pi u_{i,1}^{(k)}, \ldots, \pi u_{i,n_k}^{(k)}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{(k)}},$$

Output $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{h}_\beta^{*(0)}, \boldsymbol{t}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{h}_{\beta,i}^{*(k)}, \boldsymbol{t}_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2})$.

Let $\mathcal{B}$ be a probabilistic machine, we define the advantage of $\mathcal{B}$ for Problem 2 as follows:

$$\mathsf{Adv}_{\mathcal{B}}^{\mathrm{P2}}(\lambda) = \Big| \mathsf{Pr}\Big[\mathcal{B}(1^\lambda, \varpi) \to 1 \;\Big|\; \varpi \xleftarrow{R} \mathcal{G}_0^{\mathrm{P2}}(1^\lambda, \overrightarrow{n})\Big] \;-\; \mathsf{Pr}\Big[\mathcal{B}(1^\lambda, \varpi) \to 1 \;\Big|\; \varpi \xleftarrow{R} \mathcal{G}_1^{\mathrm{P2}}(1^\lambda, \overrightarrow{n})\Big] \Big|.$$

Lemma 2. *For any adversary $\mathcal{B}$, there exists a probabilistic machine $\mathcal{D}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{B}}^{\mathrm{P2}}(\lambda) \leq \mathsf{Adv}_{\mathcal{D}}^{\mathrm{DLIN}}(\lambda) + 5/q$.*

The proof for Lemma 2 can be found in section B.1.

Lemma 3. *For $p \in \mathbb{F}_q$, let $C_p = \{(\overrightarrow{x}, \overrightarrow{v}) | \overrightarrow{x} \cdot \overrightarrow{v} = p\} \subset V \times V^*$ where V is n-dimensional vector space $\mathbb{F}_q^n$, and V^* its dual. For all $(\overrightarrow{x}, \overrightarrow{v}) \in C_p$, for all $(\overrightarrow{r}, \overrightarrow{w}) \in C_p$, $\Pr[\overrightarrow{x}U = \overrightarrow{r} \wedge \overrightarrow{v}Z = \overrightarrow{w}] = \Pr[\overrightarrow{x}Z = \overrightarrow{r} \wedge \overrightarrow{v}U = \overrightarrow{w}] = 1/\sharp C_p$, where $Z \xleftarrow{\cup} GL(n, \mathbb{F}_q)$, $U = (Z^{-1})^T$, and $\sharp C_p$ denotes the number of elements in C_p.*

The proof of Lemma 3 was given in [28].

Proof of Theorem 1: To prove Theorem 1, We consider the following games:

Game 0. Let Game 0 denote the real security game defined in **Definition 4**.

Game 1. Game 1 is almost identical to Game 0, except that the ciphertext for challenge attribute vectors $Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)})$ and $Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)})$, challenge plaintexts $(M^{(0)}, M^{(1)})$ and a time period I is

$$\boldsymbol{c}^{(0)} = (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}}, \tag{1}$$

$$\boldsymbol{c}^{(1)} = (\delta(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d), \overrightarrow{w}_1, 0^n, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \tag{2}$$

$$\boldsymbol{c}^{(2)} = (\delta((1, -i_1), \ldots, (1, -i_\kappa)), \overrightarrow{w}_2, 0^L, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \tag{3}$$

$$\boldsymbol{c}^{(M)} = g_T^{\zeta} M^{(b)}. \tag{4}$$

where $\delta, w, \zeta, \varphi, \varphi^{(1)}, \varphi^{(2)} \xleftarrow{\cup} \mathbb{F}_q$, $b \xleftarrow{\cup} \{0, 1\}$,

$$(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d) = ((y_1^{(b)}, \ldots, y_{\mu_1}^{(b)}), \ldots, (y_{\mu_{h^{(b)}-1}+1}^{(b)}, \ldots, y_{\mu_{h^{(b)}}}^{(b)}), (y_{\mu_{h^{(b)}}+1}, \ldots, y_{\mu_{h^{(b)}+1}}), \ldots, (y_{\mu_{d-1}+1}, \ldots, y_n)), \text{and}$$

$(\overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d) \xleftarrow{\cup} \mathbb{F}_q^{\mu_{h^{(b)}+1} - \mu_{h^{(b)}}} \times \ldots \times \mathbb{F}_q^{n-\mu_{d-1}}$, $\overrightarrow{w}_1 \xleftarrow{\cup} \mathbb{F}_q^n \setminus \{\overrightarrow{0}\}$, $\overrightarrow{w}_2 \xleftarrow{\cup} \mathbb{F}_q^L \setminus \{\overrightarrow{0}\}$.

Game 2-(m, k, j') $(m = 0, \ldots, \nu - 1; k = 0, \ldots, \kappa; j = 0, \ldots, n + L)$: Game 2-$(0, 0, 0)$ is Game 1. The number of keys in $SK_{i,l} = (sk_{i,l}, \{sk_{i|_{z-1}1,l}\}_{i_z=0})$ which is a reply to the m-th delegation query, is less than or equal to $\kappa+1$, where κ is the depth of the time tree. The number of keys in $sk_{w,l} = (\boldsymbol{k}_{w,l}, \boldsymbol{k}_{w,l,\mathsf{ran},1}^{(1)}, \ldots, \boldsymbol{k}_{w,l,\mathsf{ran},l+1}^{(1)}, \boldsymbol{k}_{w,l,\mathsf{ran},1}^{(2)}, \ldots, \boldsymbol{k}_{w,l,\mathsf{ran},r+2}^{(2)}, \boldsymbol{k}_{w,l,\mathsf{del},\mu_l+1}^{(1)}, \ldots, \boldsymbol{k}_{w,l,\mathsf{del},n}^{(1)}, \boldsymbol{k}_{w,l,\mathsf{del},(2(r+1)+1)}^{(2)}, \ldots, \boldsymbol{k}_{w1,l,\mathsf{del},L}^{(2)})$, which is the k-th $sk_{w,l}$ value in $SK_{i,l}$, is less than or equal to $n + L + 1$, where $(\boldsymbol{k}_{w,l} = \boldsymbol{k}_{w,l,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(2)})$. Game 2-$(m, \kappa, n + L + 1)$ is Game 2-$(m + 1, 0, 0)$. Game 2-(m, k, j') is almost identical to Game 2-(m, k, j), except that $(m \times k \times j + 1)$-th key in $sk_{w,l}$ has the following form:

– If $j = 0$ then

$$\left.\begin{aligned}
\boldsymbol{k}_{w,l,\mathsf{dec}}^{(0)\mathsf{norm-semi}} &= (-\alpha_{\mathsf{dec}}, \epsilon, 1, \eta_{\mathsf{dec}}^{(0)}, 0)_{\mathbb{B}^{*(0)}}, \\
\boldsymbol{k}_{w,l,\mathsf{dec}}^{(1)\mathsf{norm-semi}} &= ((\alpha_{\mathsf{dec}}^{(1)} \overrightarrow{e}_1^{(1)} + \beta_{\mathsf{dec},1}^{(1)} \overrightarrow{x}_1, \ldots, \beta_{\mathsf{dec},l}^{(1)} \overrightarrow{x}_l, 0^{n-\mu_l}), (\gamma_{\mathsf{dec}}^{(1)} \overrightarrow{e}_1^{(1)} + \sigma_{\mathsf{dec},1}^{(1)} \overrightarrow{x}_1, \ldots, \\
&\qquad \sigma_{\mathsf{dec},l}^{(1)} \overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)}, \overrightarrow{\eta}_{\mathsf{dec}}^{(1)}, 0)_{\mathbb{B}^{*(1)}}, \\
\boldsymbol{k}_{w,l,\mathsf{dec}}^{(2)\mathsf{norm-semi}} &= ((\alpha_{\mathsf{dec}}^{(2)} \overrightarrow{e}_1^{(2)} + \beta_{\mathsf{dec},1}^{(2)} \overrightarrow{I}_1, \ldots, \beta_{\mathsf{dec},r}^{(2)} \overrightarrow{I}_r, 0^{L-2r}), (\gamma_{\mathsf{dec}}^{(2)} \overrightarrow{e}_1^{(2)} + \sigma_{\mathsf{dec},1}^{(2)} \overrightarrow{I}_1, \ldots, \\
&\qquad \sigma_{\mathsf{dec},r}^{(2)} \overrightarrow{I}_r, 0^{L-2r}) \cdot Z^{(2)}, \overrightarrow{\eta}_{\mathsf{dec}}^{(2)}, 0)_{\mathbb{B}^{*(2)}},
\end{aligned}\right\} \tag{5}$$

where $\epsilon, \gamma_{\mathsf{dec}}^{(1)}, \sigma_{\mathsf{dec},i}^{(1)} (i = 1, \ldots, l), \gamma_{\mathsf{dec}}^{(2)}, \sigma_{\mathsf{dec},i}^{(2)} (i = 1, \ldots, r) \xleftarrow{\cup} \mathbb{F}_q$, $Z^{(1)} \xleftarrow{\cup} GL(\mathbb{F}_q, n)$, $Z^{(2)} \xleftarrow{\cup} GL(\mathbb{F}_q, L)$, and all the other variables are generated as in Game 2-(m, k, j).

– If $j > 0$ then we have

$$\boldsymbol{k}_{(m,k,j+1)}^{\mathsf{norm-semi}} = \boldsymbol{k}_{(m,k,j+1)}^{\mathsf{normal}} + (0^n, (\sigma_{j+1,1} \overrightarrow{x}_1, \ldots, \sigma_{j+1,l} \overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)}, 0^n, 0)_{\mathbb{B}^{*(1)}} \tag{6}$$

for randomness and delegate components of the hierarchical predicate, or

$$\boldsymbol{k}^{\mathsf{norm-semi}}_{(m,k,j+1)} = \boldsymbol{k}^{\mathsf{normal}}_{(m,k,j+1)} + (0^n, (\sigma_{j+1,1}\overrightarrow{I}_1, \ldots, \sigma_{j+1,r}\overrightarrow{I}_r, 0^{L-2r}) \cdot Z^{(2)}, 0^n, 0)_{\mathbb{B}^{*(2)}} \tag{7}$$

for randomness and delegate components of the time period, where $\boldsymbol{k}^{\mathsf{normal}}_{(m,k,j+1)}$ is a correctly generated value of the reply to the j-th key in $sk_{w,l}$.

Another difference is that the challenge ciphertext has the following form:

$$\left.\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}^{(b)}_1, \ldots, \overrightarrow{y}^{(b)}_{h^{(b)}}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d), \\
&\qquad\qquad (\overrightarrow{y}^{(b)}_1, \ldots, \overrightarrow{y}^{(b)}_{h^{(b)}}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d) \cdot U^{(1)}, 0^n, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1,-i_1), \ldots, (1,-i_\kappa)), ((1,-i_1), \ldots, (1,-i_\kappa)) \cdot U^{(2)}, 0^L, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^{\zeta} M^{(b)}.
\end{aligned}\right\} \tag{8}$$

where $U^{(1)} = \left(Z^{(1)^{-1}}\right)^T$ and $U^{(2)} = \left(Z^{(2)^{-1}}\right)^T$, and all the other variables are generated as in Game 2-(m,k,j).

Game 2-$(m,k,j+1)$ $(m = 0, \ldots, \nu-1; k = 0, \ldots, \kappa; j = 0, \ldots, n+L)$: The number of keys in $SK_{i,l} = (sk_{i,l}, \{sk_{i|_{z-1}1,l}\}_{i_z=0})$ which is a reply to the m-th delegation query in the game, is less than or equal to $\kappa+1$, where κ is the depth of the time tree. The number of keys in $sk_{w,l} = (\boldsymbol{k}_{w,l}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},l+1}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},1}, \ldots, \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},r+2}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},\mu_l+1}, \ldots, \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},n}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},(2(r+1)+1)}, \ldots, \boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},L})$, which is the k-th $sk_{w,l}$ in $SK_{i,l}$, is less than or equal to $n+L+1$, where $(\boldsymbol{k}_{w,l} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}})$. Game 2-$(m,k,j+1)$ is almost identical to Game 2-(m,k,j'), except that $(j+1)$-th key in $sk_{w,l}$ is computed as follows:

– If $j = 0$ then

$$\left.\begin{aligned}
\boldsymbol{k}^{(0)\mathsf{semi}}_{w,l,\mathsf{dec}} &= (-\alpha_{\mathsf{dec}}, \epsilon, 1, \eta^{(0)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(0)}}, \\
\boldsymbol{k}^{(1)\mathsf{semi}}_{w,l,\mathsf{dec}} &= ((\alpha^{(1)}_{\mathsf{dec}}\overrightarrow{e}^{(1)}_1 + \beta^{(1)}_{\mathsf{dec},1}\overrightarrow{x}_1, \ldots, \beta^{(1)}_{\mathsf{dec},l}\overrightarrow{x}_l, 0^{n-\mu_l}), \overrightarrow{v}_1, \overrightarrow{\eta}^{(1)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(1)}}, \\
\boldsymbol{k}^{(2)\mathsf{semi}}_{w,l,\mathsf{dec}} &= ((\alpha^{(2)}_{\mathsf{dec}}\overrightarrow{e}^{(2)}_1 + \beta^{(2)}_{\mathsf{dec},1}\overrightarrow{I}_1, \ldots, \beta^{(2)}_{\mathsf{dec},r}\overrightarrow{I}_r, 0^{L-2r}), \overrightarrow{v}_2, \overrightarrow{\eta}^{(2)}_{\mathsf{dec}}, 0)_{\mathbb{B}^{*(2)}},
\end{aligned}\right\} \tag{9}$$

where $\overrightarrow{v}_1 \xleftarrow{\cup} \mathbb{F}_q^n, \overrightarrow{v}_2 \xleftarrow{\cup} \mathbb{F}_q^L$, $Z^{(1)} \xleftarrow{\cup} GL(\mathbb{F}_q, n)$, $Z^{(2)} \xleftarrow{\cup} GL(\mathbb{F}_q, L)$, and all the other variables are generated as in Game 2-$(m+1,k,j-1)$.

– If $j > 0$ then we have

$$\boldsymbol{k}^{\mathsf{semi}}_{(m,k,j+1)} = \boldsymbol{k}^{\mathsf{normal}}_{(m,k,j+1)} + (0^n, \overrightarrow{v}'_{j+1}, 0^n, 0)_{\mathbb{B}^{*(1)}} \tag{10}$$

for randomness and delegate components of the hierarchical predicate, or

$$\boldsymbol{k}^{\mathsf{semi}}_{(m,k,j+1)} = \boldsymbol{k}^{\mathsf{normal}}_{(m,k,j+1)} + (0^L, \overrightarrow{v}'_{j+1}, 0^L, 0)_{\mathbb{B}^{*(2)}} \tag{11}$$

for randomness and delegate components of the time period, where $\boldsymbol{k}^{\mathsf{normal}}_{(m+1,k,j)}$ is a correctly generated value of the reply to the j-th key in the k-th $sk_{w,l}$ in $SK_{i,l}$ and in m-th key query, and $\overrightarrow{v}'_{j+1} \xleftarrow{\cup} \mathbb{F}_q^{n_k}, k = 1, 2$.

Another difference is that the challenge ciphertext is computed as follows:

$$\left.\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}^{(b)}_1, \ldots, \overrightarrow{y}^{(b)}_{h^{(b)}}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d), \overrightarrow{w}_1, 0^n, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1,-i_1), \ldots, (1,-i_\kappa)), \overrightarrow{w}_2, 0^L, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^{\zeta} M^{(b)}.
\end{aligned}\right\} \tag{12}$$

where $\overrightarrow{w}_1 \xleftarrow{\cup} \mathbb{F}_q^n \setminus \{\overrightarrow{0}\}$, $\overrightarrow{w}_2 \xleftarrow{\cup} \mathbb{F}_q^L \setminus \{\overrightarrow{0}\}$, and all the other variables are generated as in Game 2-(m,k,j').

Game 3. Game 3 is almost identical to Game 2-$(\nu,0,0)$, except that the ciphertext for challenge attribute vectors $Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)})$ and $Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)})$, challenge plaintexts $(M^{(0)}, M^{(1)})$, and a time period I is

$$\left.\begin{aligned} \boldsymbol{c}^{(0)} &= (\delta, w, \zeta', 0, \varphi)_{\mathbb{B}^{(0)}}, \\ \boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}'_1, \ldots, \overrightarrow{y}'_d), \overrightarrow{w}_1, 0^n, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \\ \boldsymbol{c}^{(2)} &= (\delta((1,-i_1), \ldots, (1,-i_\kappa)), \overrightarrow{w}_2, 0^L, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \\ \boldsymbol{c}^{(M)} &= g_T^{\zeta} M^{(b)}. \end{aligned}\right\} \tag{13}$$

where $\zeta' \xleftarrow{\cup} \mathbb{F}_q$, $(\overrightarrow{y}'_1, \ldots, \overrightarrow{y}'_d) \xleftarrow{\cup} \mathbb{F}_q^n$, and all the other variables are generated as in Game 2-ν. We note that ζ' and $(\overrightarrow{y}'_1, \ldots, \overrightarrow{y}'_d)$ are chosen uniformly and independently from ζ and $(Y^{(0)}, Y^{(1)})$, respectively.

Let $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)$ be $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FS\text{-}HPE}}(\lambda)$ in Game 0, and $\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j))}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j'))}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)$ be the advantage of $\mathcal{A}$ in Game 1, $(2\text{-}(m,k,j))$, $(2\text{-}(m,k,j'))$, 3, respectively. It is clear that $\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda) = 0$ by Lemma 8. We will show Lemmas 4 - 7 which evaluate the gaps between pairs of $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j))}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j'))}(\lambda)$, $\mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j+1))}(\lambda)$, for $(m = 0, \ldots, \nu-1; k = 0, \ldots, \kappa; j = 0, \ldots, n+L)$ and $\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)$.

From these lemmas, we obtain

$$\begin{aligned} \mathsf{Adv}_{\mathcal{A}}^{\mathsf{FS\text{-}HPE}}(\lambda) &= \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) \\ &\leq |\, \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) \,| + \\ &\quad \sum_{m=0}^{\nu-1} \sum_{k=0}^{\kappa} \sum_{j=0}^{n+L} |\, \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j))}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j'))}(\lambda) \,| + \\ &\quad \sum_{m=0}^{\nu-1} \sum_{k=0}^{\kappa} \sum_{j=0}^{n+L} |\, \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j'))}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j+1))}(\lambda) \,| + \\ &\quad |\, \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(\nu,0,0))}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda) \,| + \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda) \\ &\leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathrm{P1}}(\lambda) + \sum_{m=0}^{\nu-1} \sum_{k=0}^{\kappa} \sum_{j=0}^{n+L} \mathsf{Adv}_{\mathcal{B}'_{2mkj}}^{\mathrm{P2}}(\lambda) + \\ &\quad \sum_{m=0}^{\nu-1} \sum_{k=0}^{\kappa} \sum_{j=0}^{n+L} \mathsf{Adv}_{\mathcal{B}_{2mk(j+1)}}^{\mathrm{P2}}(\lambda) + (10\nu(\kappa+1)(n+L+1)+1)/q \\ &\leq (2\nu(\kappa+1)(n+L+1)+1)\mathsf{Adv}_{\mathcal{D}}^{\mathrm{DLIN}}(\lambda) + (20\nu(\kappa+1)(n+L+1)+9)/q. \end{aligned}$$

This completes the proof of Theorem 1. □

Lemma 4. *For any adversary $\mathcal{A}$, there exists a probabilistic machine $\mathcal{B}_1$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $|\, \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) \,| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathrm{P1}}(\lambda)$.*

Proof. Suppose a polynomial time adversary $\mathcal{A}$ can successfully distinguish between Game 0 and Game 1. We construct a simulator $\mathcal{B}_1$ that leverages $\mathcal{A}$ as a black box to solve Problem 1. The procedure is shown as follows:

1. $\mathcal{B}_1$ is given an instance of Problem 1, i.e. $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(k)}, \widehat{\mathbb{B}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{t}_\beta^{(0)}, \{\boldsymbol{t}_{\beta,1}^{(k)}\}_{k=1,2}, \{\boldsymbol{t}_i^{(k)}\}_{i=2,\ldots,n_k,k=1,2})$ where $n_1 = n$ and $n_2 = L$, and plays the role of the challenger in the security game against adversary $\mathcal{A}$.

2. At the beginning of the game, $\mathcal{B}_1$ gives $\mathcal{A}$ the public key $PK = \big(1^\lambda, \mathsf{param}_{\overrightarrow{n}}, (\boldsymbol{b}_1^{(0)}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_5^{(0)}, \boldsymbol{b}_1^{(1)}, \ldots, \boldsymbol{b}_n^{(1)}, \boldsymbol{b}_{3n+1}^{(1)}, \boldsymbol{b}_1^{(2)}, \ldots, \boldsymbol{b}_L^{(2)}, \boldsymbol{b}_{3L+1}^{(2)}, \boldsymbol{b}_{2n+1}^{*(1)}, \ldots, \boldsymbol{b}_{3n}^{*(1)}, \boldsymbol{b}_{2L+1}^{*(2)}, \ldots, \boldsymbol{b}_{3L}^{*(2)}, \boldsymbol{b}_4^{*(0)})$, which is obtained from the Problem 1 instance.
3. When a delegation query is issued, $\mathcal{B}_1$ computes a normal secret key using Delegate, Update and $SK_{0,1}$, which is computed from $(\widehat{\mathbb{B}}^{*(0)}, \widehat{\mathbb{B}}^{*(1)}, \widehat{\mathbb{B}}^{*(2)})$.
4. When $\mathcal{B}_1$ receives challenge attribute vectors $Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)})$ and $Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)})$, challenge plaintexts $(M^{(0)}, M^{(1)})$ and a time period I from $\mathcal{A}$, $\mathcal{B}_1$ computes and returns
$$C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)}),$$
where $\boldsymbol{c}^{(0)} = \boldsymbol{t}_\beta^{(0)} + \zeta \boldsymbol{b}_3^{(0)}$, $\boldsymbol{c}^{(1)} = y_1^{(b)} \boldsymbol{t}_{\beta,1}^{(1)} + \sum_{i=2}^{\mu_{h^{(b)}}} y_i^{(b)} \boldsymbol{t}_i^{(1)} + \sum_{i=\mu_{h^{(b)}}+1}^{n} y_i \boldsymbol{t}_i^{(1)}$, $\boldsymbol{c}^{(2)} = \boldsymbol{t}_{\beta,1}^{(2)} + (-i_1)\boldsymbol{t}_2^{(2)} + \cdots + \boldsymbol{t}_{L-1}^{(2)} + (-i_\kappa)\boldsymbol{t}_L^{(2)}$, and $\boldsymbol{c}^{(M)} = g_T^\zeta M^{(b)}$ using $(\boldsymbol{t}_\beta^{(0)}, \{\boldsymbol{t}_{\beta,1}^{(k)}\}_{k=1,2}, \{\boldsymbol{t}_i^{(k)}\}_{i=2,\ldots,n_k;k=1,2}, \boldsymbol{b}_3^{(0)})$ from the instance of Problem 1 and $(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)})$, $M^{(b)}$, I where $\zeta, y_{\mu_{h^{(b)}}+1}, \ldots, y_n \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $b \xleftarrow{\mathsf{U}} \{0,1\}$. $i_1, \ldots, i_\kappa$ are parsed from I.
5. After the challenge phase, delegation oracle simulation for a key query is executed in the same manner as step 3.
6. $\mathcal{A}$ outputs a bit b'. If $b = b'$, $\mathcal{B}_1$ outputs 1. Otherwise, $\mathcal{B}_1$ outputs 0.

Claim. For $\beta = 0$ the challenge ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$ generated in step 4 is distributed exactly as in Game 0, whereas if $\beta = 1$, the challenge ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$ generated in step 4 is identically distributed to Game 1.

Proof. First recall that $y_1^{(b)} = 1$. If $\beta = 0$ then the ciphertext given by

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, 0, \zeta, 0, \rho)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_d), 0^{2n}, \rho^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1, -i_1), \ldots, (1, -i_\kappa)), 0^{2L}, \rho^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^\zeta M.
\end{aligned}$$

is the challenge ciphertext from Game 0. In contrast, if $\beta = 1$ then the following components of the ciphertext have a different form

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, u, \zeta, 0, \rho)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}_1, \ldots, \overrightarrow{y}_d), \overrightarrow{u}^{(1)}, 0^n, \rho^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1, -i_1), \ldots, (1, -i_\kappa)), \overrightarrow{u}^{(2)}, 0^L, \rho^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^\zeta M^{(b)}.
\end{aligned}$$

the challenge ciphertext from Game 1.

From the above claim, if $\beta = 0$ then simulated ciphertexts are distributed exactly as in Game 0, whereas for $\beta = 1$ their distribution is identical to Game 1. Therefore, $|\ \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)\ | = \left| \mathsf{Pr}\Big[\mathcal{B}_1(1^\lambda, \varpi) \to 1 \;\Big|\; \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathrm{P1}}(1^\lambda, \overrightarrow{n})\Big] - \mathsf{Pr}\Big[\mathcal{B}_1(1^\lambda, \varpi) \to 1 \;\Big|\; \varpi \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathrm{P1}}(1^\lambda, \overrightarrow{n})\Big] \right| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathrm{P1}}(\lambda)$. This completes the proof of **Lemma 4**. □

Lemma 5. *For any adversary $\mathcal{A}$, there exists a probabilistic machine $\mathcal{B}'_{2mkj}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ,* $|\ \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j))}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j'))}(\lambda)\ | \leq \mathsf{Adv}_{\mathcal{B}'_{2mkj}}^{\mathrm{P2}}(\lambda) + 5/q$.

Proof. Suppose a polynomial time adversary $\mathcal{A}$ can successfully distinguish between Game 2-(m,k,j) and Game 2-(m,k,j'). We construct a simulator $\mathcal{B}'_{2mkj}$ that leverages $\mathcal{A}$ as a black box to solve Problem 2. The procedure is shown as follows:

1. $\mathcal{B}'_{2mkj}$ is given an instance of Problem 2, that is a tuple $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{h}^{*(0)}_{\beta}, \boldsymbol{t}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{h}^{*(k)}_{\beta,i}, \boldsymbol{t}^{(k)}_{i}\}_{i=1,\ldots,n_k}\}_{k=1,2})$ where $n_1 = n$ and $n_2 = L$, and acts as challenger in the security game against adversary $\mathcal{A}$.
2. At the beginning of the game, $\mathcal{B}'_{2mkj}$ gives $\mathcal{A}$ the public key $PK = \big(1^{\lambda}, \mathsf{param}_{\overrightarrow{n}}, (\boldsymbol{b}^{(0)}_1, \boldsymbol{b}^{(0)}_3, \boldsymbol{b}^{(0)}_5, \boldsymbol{b}^{(1)}_1, \ldots, \boldsymbol{b}^{(1)}_n, \boldsymbol{b}^{(1)}_{3n+1}, \boldsymbol{b}^{(2)}_1, \ldots, \boldsymbol{b}^{(2)}_L, \boldsymbol{b}^{(2)}_{3L+1}, \boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n}, \boldsymbol{b}^{*(2)}_{2L+1}, \ldots, \boldsymbol{b}^{*(2)}_{3L}, \boldsymbol{b}^{*(0)}_4)$, which is obtained from the Problem 2 instance.
3. The answer to the s-th key in the k-th $sk_{w,l}$ in the m-th $SK_{i,l}$ query for time period i and hierarchical predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ is as follows:
   a) For $0 \leq s \leq j$ the algorithm $\mathcal{B}'_{2mkj}$ computes a semi-functional key using $\{\mathbb{B}^{*(k)}\}_{k=0,1,2}$ of the Problem 2 instance.
   b) For $s = j + 1$ it computes as follows:
   – If $j = 0$ then it computes

$$(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}}, \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}})$$

using $\{\boldsymbol{h}^{*(0)}_{\beta}, \boldsymbol{b}^{*(0)}_1, \boldsymbol{b}^{*(0)}_3, \{\boldsymbol{h}^{*(i)}_{\beta,j}, \boldsymbol{b}^{*(i)}_j\}_{i=1,2;j=1,\ldots,n_i}\}$ of the Problem 2 instance as follows:

$$\text{For } i = 1, 2: \ \varrho_i, v_i, v'_i, \theta_i, \vartheta_k (k = 1, \ldots, l), \phi_{k'} (k' = 1, \ldots, r) \xleftarrow{\mathsf{U}} \mathbb{F}_q;$$

$$\boldsymbol{s}^{(0)}_{\beta} = \sum_{i=1}^{2} (\varrho_i \boldsymbol{h}^{*(0)}_{\beta} + v_i \boldsymbol{b}^{*(0)}_1), \ \ \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} = -\boldsymbol{s}^{(0)}_{\beta} + \boldsymbol{b}^{*(0)}_3,$$

$$\text{For } i = 1, 2 \text{ and } j = 1, \ldots, n_i:$$

$$\boldsymbol{s}^{(i)}_{\beta,j} = \theta_i \boldsymbol{h}^{*(i)}_{\beta,j} + v'_i \boldsymbol{b}^{*(i)}_j, \ \ \widehat{\boldsymbol{s}}^{(i)}_{\beta,j} = \varrho_i \boldsymbol{h}^{*(i)}_{\beta,j} + v_i \boldsymbol{b}^{*(i)}_j,$$

$$\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} = \sum_{k=1}^{l} \left(\vartheta_k \sum_{j=\mu_{k-1}+1}^{\mu_k} x_j \boldsymbol{s}^{(1)}_{\beta,j} \right) + \widehat{\boldsymbol{s}}^{(1)}_{\beta,1},$$

$$\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} = \sum_{k'=1}^{r} \left(\phi_{k'} \sum_{j=2k'-1}^{2k'} I_j \boldsymbol{s}^{(2)}_{\beta,j} \right) + \widehat{\boldsymbol{s}}^{(2)}_{\beta,1},$$

– If $j > 0$ then compute randomness and delegation components using $\{\{\boldsymbol{h}^{*(i)}_{\beta,j}, \boldsymbol{b}^{*(i)}_j\}_{i=1,2;j=1,\ldots,n_i}\}$ of the Problem 2 instance as follows::

$$\text{For } i = 1, 2: \ v'_i, \theta_i, \psi, \psi', \vartheta_k (k = 1, \ldots, l), \phi_{k'} (k' = 1, \ldots, r) \xleftarrow{\mathsf{U}} \mathbb{F}_q;$$

$$\text{For } i = 1, 2 \text{ and } j = 1, \ldots, n_i:$$

$$\boldsymbol{s}^{(i)}_{\beta,j} = \theta_i \boldsymbol{h}^{*(i)}_{\beta,j} + v'_i \boldsymbol{b}^{*(i)}_j,$$

$$\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran,h}} = \sum_{k=1}^{l} \left(\vartheta_k \sum_{j=\mu_{k-1}+1}^{\mu_k} x_j \boldsymbol{s}^{(1)}_{\beta,j} \right),$$

$$\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran,h'}} = \sum_{k'=1}^{r} \left(\phi_{k'} \sum_{j=2k'-1}^{2k'} I_j \boldsymbol{s}^{(2)}_{\beta,j} \right),$$

$$\boldsymbol{k}^{(1)}_{w,l,\mathsf{del,h}} = \sum_{k=1}^{l} \left(\vartheta_k \sum_{j=\mu_{k-1}+1}^{\mu_k} x_j \boldsymbol{s}^{(1)}_{\beta,j} \right) + \psi \boldsymbol{b}^{*(1)}_h,$$

$$\boldsymbol{k}^{(2)}_{w,l,\mathsf{del,h'}} = \sum_{k'=1}^{r} \left(\phi_{k'} \sum_{j=2k'-1}^{2k'} I_j \boldsymbol{s}^{(2)}_{\beta,j} \right) + \psi' \boldsymbol{b}^{*(2)}_{h'},$$

c) For $s \geq j+2$ the algorithm $\mathcal{B}'_{2mkj}$ computes a normal key using $\{\mathbb{B}^{*(k)}\}_{k=0,1,2}$ from the Problem 2 instance.

4. When $\mathcal{B}'_{2mkj}$ receives challenge attribute vectors $Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)})$ and $Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)})$, challenge plaintexts $(M^{(0)}, M^{(1)})$ and a time period I from $\mathcal{A}$, $\mathcal{B}'_{2mkj}$ computes and returns the ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$ where

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= \boldsymbol{t}^{(0)} + \zeta \boldsymbol{b}_3^{(0)} + \varphi \boldsymbol{b}_5^{(0)}, \\
\boldsymbol{c}^{(1)} &= \sum_{i=1}^{\mu_{h^{(b)}}} y_i^{(b)} \boldsymbol{t}_i^{(1)} + \sum_{i=\mu_{h^{(b)}}+1}^{n} y_i \boldsymbol{t}_i^{(1)} + \varphi^{(1)} \boldsymbol{b}_{3n+1}^{(1)}, \\
\boldsymbol{c}^{(2)} &= \sum_{j=1}^{L} I_j \boldsymbol{t}_j^{(2)} + \varphi^{(2)} \boldsymbol{b}_{3L+1}^{(2)}, \\
\boldsymbol{c}^{(M)} &= g_T^{\zeta} M^{(b)},
\end{aligned}$$

using $(\boldsymbol{t}^{(0)}, \{\boldsymbol{t}_i^{(1)}\}_{i=1,\ldots,n}, \{\boldsymbol{t}_i^{(2)}\}_{i=1,\ldots,L}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_5^{(0)}, \boldsymbol{b}_{3n+1}^{(1)}, \boldsymbol{b}_{3L+1}^{(2)})$ from the instance of Problem 2 and $(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)})$, $M^{(b)}$, I where $\zeta, \varphi, \varphi^{(1)}, \varphi^{(2)}, y_{\mu_{h^{(b)}}+1}, \ldots, y_n \xleftarrow{\cup} \mathbb{F}_q$, $b \xleftarrow{\cup} \{0,1\}$. (I_{2k-1}, I_{2k}) denotes the vector for i_k, and $i_1, \ldots, i_\kappa$ is parsed from I.

5. After the challenge phase, delegation oracle simulation for a key query is executed in the same manner as step 3.
6. $\mathcal{A}$ outputs a bit b'. If $b = b'$, $\mathcal{B}'_{2mkj}$ outputs 1. Otherwise, $\mathcal{B}'_{2mkj}$ outputs 0.

Claim. The distribution of the view of adversary $\mathcal{A}$ in the above-mentioned game simulated by $\mathcal{B}'_{2mkj}$ given a Problem 2 instance with $\beta \in \{0,1\}$ is the same as that in Game 2-(m,k,j) (resp. Game 2-(m,k,j')) if $\beta = 0$ (resp. $\beta = 1$) except with probability $4/q$ (resp. $1/q$).

Proof. It is clear that $\mathcal{B}'_{2mkj}$'s simulation of the public key generation (step 2) and the answers to the s-th query where $s \neq j+1$ (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and delegation oracle in Game 2-(m,k,j) and Game 2-(m,k,j').

Next we analyze the distribution of the s-th key in the k-th $sk_{w,l}$ in the m-th $SK_{i,l}$ query for time period i and hierarchical predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ where $s = j+1$ (case (b) of steps (3) and (5)). In this case values $\boldsymbol{s}_\beta^{(0)}, \boldsymbol{s}_{\beta,j}^{(i)}, \widehat{\boldsymbol{s}}_{\beta,j}^{(i)}$, $i = 1,2$, $j = 1, \ldots, n_i$ can be expressed as follows. Let $\beta^{(i)} = \theta_i \omega + v'_i$, $\alpha^{(i)} = \varrho_i \omega + v_i$, $\alpha = \alpha^{(1)} + \alpha^{(2)}$, $\gamma = \varrho_1 + \varrho_2$, $\epsilon = \gamma z$. Then,

$$\begin{aligned}
\boldsymbol{s}_0^{(0)} &= (\alpha, 0, 0, \gamma\xi, 0)_{\mathbb{B}^{*(0)}}, & \boldsymbol{s}_1^{(0)} &= (\alpha, \epsilon, 0, \gamma\xi, 0)_{\mathbb{B}^{*(0)}}, \\
\boldsymbol{s}_{0,j}^{(i)} &= (\overbrace{\beta^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{0^{n_i}}^{n_i}, \overbrace{\theta_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}}, & \boldsymbol{s}_{1,j}^{(i)} &= (\overbrace{\beta^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{\theta_i \overrightarrow{z}_j^{(i)}}^{n_i}, \overbrace{\theta_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}}, \\
\widehat{\boldsymbol{s}}_{0,j}^{(i)} &= (\overbrace{\alpha^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{0^{n_i}}^{n_i}, \overbrace{\varrho_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}}, & \widehat{\boldsymbol{s}}_{1,j}^{(i)} &= (\overbrace{\alpha^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{\varrho_i \overrightarrow{z}_j^{(i)}}^{n_i}, \overbrace{\varrho_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}},
\end{aligned}$$

where $\overrightarrow{z}_j^{(i)} = z_{j,1}^{(i)}, \ldots, z_{j,n_i}^{(i)}$, $\omega, z, \xi, \{\overrightarrow{\omega}_j^{(i)}, \overrightarrow{z}_j^{(i)}\}_{i=1,2;j=1,\ldots,n_i}$ are defined as in Problem 2. If $\beta = 1$ in the instance of Problem 2 then the decryption component $(\boldsymbol{k}_{w,l,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(2)})$ has the same distribution as in Eq. 5, except that $\epsilon w = \gamma$, where $\gamma = \varrho_1 + \varrho_2$ and $w = u \xleftarrow{\cup} \mathbb{F}_q$ of $\boldsymbol{c}_0$ in Eq. 1. Randomness and delegation components for the hierarchical predicate and time period have the same distribution as in Eq. 6 and 7 respectively.

Next, we show that the joint distribution of the response to $j+1$-th key in the k-th $sk_{w,l}$ in the m-th $SK_{i,l}$ query and of the challenge ciphertext in the simulation by $\mathcal{B}'_{2mkj}$ for the given instance of Problem 2 is equivalent to the distribution in Game 2-(m,k,j) if $\beta = 0$ and to the distribution in Game 2-(m,k,j') if $\beta = 1$.

If $\beta = 0$ then this equivalence follows easily, unless one of the following conditions holds: (1) ω defined in Problem 2 is zero, (2) $w = 0$, (3) $\overrightarrow{w}_1 = \overrightarrow{0}$, (4) $\overrightarrow{w}_2 = \overrightarrow{0}$. However, those events occur with probability $4/q$.

If $\beta = 1$, then $\mathcal{B}'_{2mkj}$'s simulation for the key is the same as that expressed in Eq. 5, 6 and 7, and $\mathcal{B}'_{2mkj}$'s simulation for the challenge ciphertext is the same as that expressed in Eq. 8, except that $\epsilon w = \gamma$, where $\gamma = \varrho_1 + \varrho_2$, and $w \xleftarrow{\mathsf{U}} \mathbb{F}_q$ of $\boldsymbol{c}_0$ in Eq. 1.

Therefore, we will show that γ is uniformly distributed and is independent from the other variables used in the simulation by $\mathcal{B}'_{2mkj}$. Since γ is related to $\overrightarrow{A}_1, \overrightarrow{A}_2, \overrightarrow{B}_1$, and $\overrightarrow{B}_2$, where $\overrightarrow{A}_1 = (\varrho_1 \overrightarrow{e}_1^{(1)} + \theta'_1 \overrightarrow{x}_1, \ldots, \theta'_l \overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)}$, $\overrightarrow{A}_2 = (\varrho_2 \overrightarrow{e}_1^{(2)} + \theta''_1 \overrightarrow{I}_1, \ldots, \theta''_\kappa \overrightarrow{I}_\kappa) \cdot Z^{(2)}$, and $\overrightarrow{B}_1 = (\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d) \cdot U^{(1)}$, $\overrightarrow{B}_2 = (\overrightarrow{I}'_1, \ldots, \overrightarrow{I}'_\kappa) \cdot U^{(2)}$ where $\overrightarrow{I}_z = (1, -i_z)$ and $i_1, \ldots, i_\kappa$ is parsed from I. We analyze joint distribution of these variables for the cases that appear in **Definition 4**.

1. When $i > I$, i.e., the time period of the queried key is after the time period of encoded in the ciphertext, due to Lemma 3, the pair $(\overrightarrow{A}_2, \overrightarrow{B}_2)$ is uniformly and independently distributed over $C_{\sum_{z=1}^{\kappa} \theta''_z \cdot (\overrightarrow{I}'_z \cdot \overrightarrow{I}_z) + \varrho_2} = \{(\overrightarrow{w}, \overrightarrow{r}) | \overrightarrow{w} \cdot \overrightarrow{r} = \sum_{z=1}^{\kappa} \theta''_z \cdot (\overrightarrow{I}'_z \cdot \overrightarrow{I}_z) + \varrho_2\}$ (over $Z^{(2)} \xleftarrow{\mathsf{U}} GL(\mathbb{F}_q, n)$). Since $\theta''_z \xleftarrow{\mathsf{U}} \mathbb{F}_q$, the pair $(\overrightarrow{A}_2, \overrightarrow{B}_2)$ is thus uniformly and independently distributed over $\mathbb{F}_q^{2n}$.
2. When $i \leq I$ and $f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}(\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)}) = f_{(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)}(\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)}) = 0$, the pair $(\overrightarrow{A}_2, \overrightarrow{B}_2)$ is uniformly and independently distributed over C_{ϱ_2}(over $Z^{(2)} \xleftarrow{\mathsf{U}} GL(\mathbb{F}_q, n)$). The pair $(\overrightarrow{A}_1, \overrightarrow{B}_1)$ is uniformly and independently distributed over $C_{\sum_{z=1}^{l} \theta'_z \cdot (\overrightarrow{x}_z \cdot \overrightarrow{y}_z) + \varrho_1}$ (over $Z^{(1)} \xleftarrow{\mathsf{U}} GL(\mathbb{F}_q, L)$). Since $\theta'_z \xleftarrow{\mathsf{U}} \mathbb{F}_q$, the pair $(\overrightarrow{A}_1, \overrightarrow{B}_1)$ is thus uniformly and independently distributed over $\mathbb{F}_q^{2L}$.

Considering the adversary $\mathcal{A}$'s restriction on key queries from **Definition 4**, in above two cases at least one of $(\overrightarrow{A}_1, \overrightarrow{B}_1$ and $(\overrightarrow{A}_2, \overrightarrow{B}_2)$ is uniformly and independently distributed over $\mathbb{F}_q^{2n_k}$ for $k = 1, 2$. Therefore, $\gamma = \varrho_1 + \varrho_2$ is independent from the distribution of ϱ_1(resp. ϱ_2), which can be given by $(\overrightarrow{A}_1, \overrightarrow{B}_1)$ (resp. $(\overrightarrow{A}_2, \overrightarrow{B}_2)$). Thus, γ is uniformly and independently distributed from the other variables in the simulation of $\mathcal{B}'_{2mkj}$.

Therefore, the view of $\mathcal{A}$ in the game simulated by $\mathcal{B}'_{2mkj}$ on input an instance of Problem 2 with $\beta = 1$ is the same as in Game 2-(m, k, j') unless $\omega = 0$ occurs. This event happens with probability $1/q$.

This completes the proof of **Lemma 5**. □

Lemma 6. *For any adversary $\mathcal{A}$, there exists a probabilistic machine $\mathcal{B}_{2mk(j+1)}$, whose running time is essentially the same as that of $\mathcal{A}$, such that for any security parameter λ, $| \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j'))}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2\text{-}(m,k,j+1))}(\lambda) | \leq \mathsf{Adv}^{\mathrm{P2}}_{\mathcal{B}_{2mk(j+1)}}(\lambda) + 5/q$.*

Proof. Suppose a polynomial time adversary $\mathcal{A}$ can successfully distinguish between Game 2-(m, k, j') and Game 2-$(m, k, j+1)$. We construct a simulator $\mathcal{B}_{2mk(j+1)}$ that leverages $\mathcal{A}$ as a black box to solve Problem 2. The procedure is shown as follows:

1. $\mathcal{B}_{2mk(j+1)}$ is given an instance of Problem 2, that is a tuple $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{h}_\beta^{*(0)}, \boldsymbol{t}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{h}_{\beta,i}^{*(k)}, \boldsymbol{t}_i^{(k)}\}_{i=1,\ldots,n_k}\}_{k=1,2})$ where $n_1 = n$ and $n_2 = L$, and acts as challenger in the security game against adversary $\mathcal{A}$.
2. At the beginning of the game, $\mathcal{B}_{2mk(j+1)}$ gives $\mathcal{A}$ the public key $PK = (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, (\boldsymbol{b}_1^{(0)}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_5^{(0)}, \boldsymbol{b}_1^{(1)}, \ldots, \boldsymbol{b}_n^{(1)}, \boldsymbol{b}_{3n+1}^{(1)}, \boldsymbol{b}_1^{(2)}, \ldots, \boldsymbol{b}_L^{(2)}, \boldsymbol{b}_{3L+1}^{(2)}, \boldsymbol{b}_{2n+1}^{*(1)}, \ldots, \boldsymbol{b}_{3n}^{*(1)}, \boldsymbol{b}_{2L+1}^{*(2)}, \ldots, \boldsymbol{b}_{3L}^{*(2)}, \boldsymbol{b}_4^{*(0)})$, which is obtained from the Problem 2 instance.
3. The answer to the s-th key in the k-th $sk_{w,l}$ in the m-th $SK_{i,l}$ query for time period i and hierarchical predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ is as follows:
   a) For $0 \leq s \leq j$ the algorithm $\mathcal{B}_{2mk(j+1)}$ computes a semi-functional key using $\{\mathbb{B}^{*(k)}\}_{k=0,1,2}$ of the Problem 2 instance.
   b) For $s = j + 1$ it proceeds as follows:
      – If $j = 0$ then it computes

$$(\boldsymbol{k}_{w,l,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(2)})$$

using $\{\boldsymbol{h}_{\beta}^{*(0)}, \boldsymbol{b}_1^{*(0)}, \boldsymbol{b}_3^{*(0)}, \{\boldsymbol{h}_{\beta,j}^{*(i)}, \boldsymbol{b}_j^{*(i)}\}_{i=1,2;j=1,\ldots,n_i}\}$ of the Problem 2 instance as follows:

$$\text{For } i=1,2: \ \varrho_i, v_i, v_i', \theta_i, r', \vartheta_k(k=1,\ldots,l), \phi_{k'}(k'=1,\ldots,r) \xleftarrow{\mathsf{U}} \mathbb{F}_q;$$

$$\boldsymbol{s}_{\beta}^{(0)} = \sum_{i=1}^{2}(\varrho_i \boldsymbol{h}_{\beta}^{*(0)} + v_i \boldsymbol{b}_1^{*(0)}), \ \ \boldsymbol{k}_{w,l,\mathsf{dec}}^{(0)} = -\boldsymbol{s}_{\beta}^{(0)} + r'\boldsymbol{b}_2^{*(0)} + \boldsymbol{b}_3^{*(0)},$$

$$\text{For } i=1,2 \text{ and } j=1,\ldots,n_i:$$

$$\boldsymbol{s}_{\beta,j}^{(i)} = \theta_i \boldsymbol{h}_{\beta,j}^{*(i)} + v_i' \boldsymbol{b}_j^{*(i)}, \ \ \widehat{\boldsymbol{s}}_{\beta,j}^{(i)} = \varrho_i \boldsymbol{h}_{\beta,j}^{*(i)} + v_i \boldsymbol{b}_j^{*(i)},$$

$$\boldsymbol{k}_{w,l,\mathsf{dec}}^{(1)} = \sum_{k=1}^{l}\left(\vartheta_k \sum_{j=\mu_{k-1}+1}^{\mu_k} x_j \boldsymbol{s}_{\beta,j}^{(1)}\right) + \widehat{\boldsymbol{s}}_{\beta,1}^{(1)} + \sum_{k=1}^{n} v_k \boldsymbol{b}_{n+k}^{*(1)},$$

$$\boldsymbol{k}_{w,l,\mathsf{dec}}^{(2)} = \sum_{k'=1}^{r}\left(\phi_{k'} \sum_{j=2k'-1}^{2k'} I_j \boldsymbol{s}_{\beta,j}^{(2)}\right) + \widehat{\boldsymbol{s}}_{\beta,1}^{(2)} + \sum_{k'=1}^{L} v_{k'}' \boldsymbol{b}_{L+k'}^{*(2)},$$

where $v_k(k=1,\ldots,n), v_{k'}'(k'=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$.

$-$ If $j>0$ then compute randomness and delegation components using $\{\{\boldsymbol{h}_{\beta,j}^{*(i)}, \boldsymbol{b}_j^{*(i)}\}_{i=1,2;j=1,\ldots,n_i}\}$ of the Problem 2 instance as follows::

$$\text{For } i=1,2: \ v_i', \theta_i, \psi, \psi', \vartheta_k(k=1,\ldots,l), \phi_{k'}(k'=1,\ldots,r) \xleftarrow{\mathsf{U}} \mathbb{F}_q;$$

$$\text{For } i=1,2 \text{ and } j=1,\ldots,n_i:$$

$$\boldsymbol{s}_{\beta,j}^{(i)} = \theta_i \boldsymbol{h}_{\beta,j}^{*(i)} + v_i' \boldsymbol{b}_j^{*(i)},$$

$$\boldsymbol{k}_{w,l,\mathsf{ran,h}}^{(1)} = \sum_{k=1}^{l}\left(\vartheta_k \sum_{j=\mu_{k-1}+1}^{\mu_k} x_j \boldsymbol{s}_{\beta,j}^{(1)}\right) + \sum_{k=1}^{n} v_k \boldsymbol{b}_{n+k}^{*(1)},$$

$$\boldsymbol{k}_{w,l,\mathsf{ran,h}'}^{(2)} = \sum_{k'=1}^{r}\left(\phi_{k'} \sum_{j=2k'-1}^{2k'} I_j \boldsymbol{s}_{\beta,j}^{(2)}\right) + \sum_{k'=1}^{L} v_{k'}' \boldsymbol{b}_{L+k'}^{*(2)},$$

$$\boldsymbol{k}_{w,l,\mathsf{del,h}}^{(1)} = \sum_{k=1}^{l}\left(\vartheta_k \sum_{j=\mu_{k-1}+1}^{\mu_k} x_j \boldsymbol{s}_{\beta,j}^{(1)}\right) + \psi \boldsymbol{b}_h^{*(1)} + \sum_{k=1}^{n} v_k \boldsymbol{b}_{n+k}^{*(1)},$$

$$\boldsymbol{k}_{w,l,\mathsf{del,h}'}^{(2)} = \sum_{k'=1}^{r}\left(\phi_{k'} \sum_{j=2k'-1}^{2k'} I_j \boldsymbol{s}_{\beta,j}^{(2)}\right) + \psi' \boldsymbol{b}_{h'}^{*(2)} + \sum_{k'=1}^{L} v_{k'}' \boldsymbol{b}_{L+k'}^{*(2)},$$

where $v_k(k=1,\ldots,n), v_{k'}'(k'=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$.

c) For $s \geq j+2$ the algorithm $\mathcal{B}_{2mk(j+1)}$ computes a normal key using $\{\mathbb{B}^{*(k)}\}_{k=0,1,2}$ from the instance of Problem 2.

4. When $\mathcal{B}_{2mk(j+1)}$ receives challenge attribute vectors $Y^{(0)} = (\overrightarrow{y}_1^{(0)}, \ldots, \overrightarrow{y}_{h^{(0)}}^{(0)})$ and $Y^{(1)} = (\overrightarrow{y}_1^{(1)}, \ldots, \overrightarrow{y}_{h^{(1)}}^{(1)})$, challenge plaintexts $(M^{(0)}, M^{(1)})$ and a time period I from $\mathcal{A}$, $\mathcal{B}_{2mk(j+1)}$ computes and returns the ciphertext $C = (\boldsymbol{c}^{(0)}, \boldsymbol{c}^{(1)}, \boldsymbol{c}^{(2)}, \boldsymbol{c}^{(M)})$ where

$$\boldsymbol{c}^{(0)} = \boldsymbol{t}^{(0)} + \zeta \boldsymbol{b}_3^{(0)} + \varphi \boldsymbol{b}_5^{(0)},$$

$$\boldsymbol{c}^{(1)} = \sum_{i=1}^{\mu_{h^{(b)}}} y_i^{(b)} \boldsymbol{t}_i^{(1)} + \sum_{i=\mu_{h^{(b)}}+1}^{n} y_i \boldsymbol{t}_i^{(1)} + \varphi^{(1)} \boldsymbol{b}_{3n+1}^{(1)},$$

$$\boldsymbol{c}^{(2)} = \sum_{j=1}^{L} I_j \boldsymbol{t}_j^{(2)} + \varphi^{(2)} \boldsymbol{b}_{3L+1}^{(2)},$$
$$\boldsymbol{c}^{(M)} = g_T^{\zeta} M^{(b)},$$

using $(\boldsymbol{t}^{(0)}, \{\boldsymbol{t}_i^{(1)}\}_{i=1,\ldots,n}, \{\boldsymbol{t}_i^{(2)}\}_{i=1,\ldots,L}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_5^{(0)}, \boldsymbol{b}_{3n+1}^{(1)}, \boldsymbol{b}_{3L+1}^{(2)})$ from the instance of Problem 2 and $(\overrightarrow{y}_1^{(b)}, \ldots, \overrightarrow{y}_{h^{(b)}}^{(b)})$, $M^{(b)}$, I where $\zeta, \varphi, \varphi^{(1)}, \varphi^{(2)}, y_{\mu_{h^{(b)}+1}}, \ldots, y_n \xleftarrow{\cup} \mathbb{F}_q$, $b \xleftarrow{\cup} \{0,1\}$. (I_{2k-1}, I_{2k}) denotes the vector for i_k, and $i_1, \ldots, i_\kappa$ is parsed from I.

5. After the challenge phase, delegation oracle simulation for a key query is executed in the same manner as step 3.
6. $\mathcal{A}$ outputs a bit b'. If $b = b'$, $\mathcal{B}_{2mk(j+1)}$ outputs 1. Otherwise, $\mathcal{B}_{2mk(j+1)}$ outputs 0.

Claim. The distribution of the view of adversary $\mathcal{A}$ in the above-mentioned game simulated by $\mathcal{B}_{2mk(j+1)}$ given a Problem 2 instance with $\beta \in \{0,1\}$ is the same as that in Game 2-$(m,k,j+1)$ (resp. Game 2-(m,k,j')) if $\beta = 0$ (resp. $\beta = 1$) except with probability $4/q$ (resp. $1/q$).

Proof. It is clear that $\mathcal{B}_{2mk(j+1)}$'s simulation of the public key generation (step 2) and the answers to the s-th query where $s \neq j+1$ (case (a) and (c) of steps (3) and (5)) are exactly the same as the Setup and delegation oracle in Game 2-$(m,k,j+1)$ and Game 2-(m,k,j').

Next we analyze the distribution of the s-th key in the k-th $sk_{w,l}$ in the m-th $SK_{i,l}$ query for time period i and hierarchical predicate vectors $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$ where $s = j+1$ (case (b) of steps (3) and (5)). In this case values $\boldsymbol{s}_\beta^{(0)}, \boldsymbol{s}_{\beta,j}^{(i)}, \widehat{\boldsymbol{s}}_{\beta,j}^{(i)}$, $i = 1,2$, $j = 1, \ldots, n_i$ can be expressed as follows. Let $\beta^{(i)} = \theta_i \omega + v_i'$, $\alpha^{(i)} = \varrho_i \omega + v_i$, $\alpha = \alpha^{(1)} + \alpha^{(2)}$, $\gamma = \varrho_1 + \varrho_2$. Then,

$$\boldsymbol{s}_0^{(0)} = (\alpha, 0, 0, \gamma\xi, 0)_{\mathbb{B}^{*(0)}}, \qquad \boldsymbol{s}_1^{(0)} = (\alpha, \epsilon, 0, \gamma\xi, 0)_{\mathbb{B}^{*(0)}},$$
$$\boldsymbol{s}_{0,j}^{(i)} = (\overbrace{\beta^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{0^{n_i}}^{n_i}, \overbrace{\theta_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}}, \quad \boldsymbol{s}_{1,j}^{(i)} = (\overbrace{\beta^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{\theta_i \overrightarrow{z}_j^{(i)}}^{n_i}, \overbrace{\theta_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}},$$
$$\widehat{\boldsymbol{s}}_{0,j}^{(i)} = (\overbrace{\alpha^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{0^{n_i}}^{n_i}, \overbrace{\varrho_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}}, \quad \widehat{\boldsymbol{s}}_{1,j}^{(i)} = (\overbrace{\alpha^{(i)}\overrightarrow{e}_j^{(i)}}^{n_i}, \overbrace{\varrho_i \overrightarrow{z}_j^{(i)}}^{n_i}, \overbrace{\varrho_i \overrightarrow{\omega}_j^{(i)}}^{n_i}, \overbrace{0}^{1})_{\mathbb{B}^{*(i)}},$$

where $\overrightarrow{z}_j^{(i)} = z_{j,1}^{(i)}, \ldots, z_{j,n_i}^{(i)}, \omega, \xi, \{\overrightarrow{\omega}_j^{(i)}, \overrightarrow{z}_j^{(i)}\}_{i=1,2;j=1,\ldots,n_i}$ are defined as in Problem 2. If $\beta = 1$ in the instance of Problem 2 then the decryption component $(\boldsymbol{k}_{w,l,\mathsf{dec}}^{(0)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(1)}, \boldsymbol{k}_{w,l,\mathsf{dec}}^{(2)})$ has the same distribution as in Eq. 5, except that $(\gamma_{\mathsf{dec}}^{(1)}\overrightarrow{e}_1^{(1)} + \sigma_{\mathsf{dec},1}^{(1)}\overrightarrow{x}_1, \ldots, \sigma_{\mathsf{dec},l}^{(1)}\overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)} + \overrightarrow{v}_1'$ and $(\gamma_{\mathsf{dec}}^{(2)}\overrightarrow{e}_1^{(2)} + \sigma_{\mathsf{dec},1}^{(2)}\overrightarrow{I}_1, \ldots, \sigma_{\mathsf{dec},r}^{(2)}\overrightarrow{I}_r, 0^{L-2r}) \cdot Z^{(2)} + \overrightarrow{v}_2'$ where $\overrightarrow{v}_1' \xleftarrow{\cup} \mathbb{F}_q^n, \overrightarrow{v}_2' \xleftarrow{\cup} \mathbb{F}_q^L$. Randomness and delegation components for the hierarchical predicate and time period have the same distribution as in Eq. 6 and 7 respectively, except the added $\overrightarrow{v}_1'$ and $\overrightarrow{v}_2'$ randomness.

Next, we show that the joint distribution of the response to j-th key in the k-th $sk_{w,l}$ in the m-th $SK_{i,l}$ query and of the challenge ciphertext in the simulation by $\mathcal{B}_{2mk(j+1)}$ for the given instance of Problem 2 is equivalent to the distribution in Game 2-$(m,k,j+1)$ if $\beta = 0$ and to the distribution in Game 2-(m,k,j') if $\beta = 1$.

If $\beta = 0$ then this equivalence follows easily, unless one of the following conditions holds: (1) ω defined in Problem 2 is zero, (2) $w = 0$, (3) $\overrightarrow{w}_1 = \overrightarrow{0}$, (4) $\overrightarrow{w}_2 = \overrightarrow{0}$. However, those events occur with probability $4/q$.

If $\beta = 1$, then $\mathcal{B}_{2mk(j+1)}$'s simulation for the key is the same as that expressed in Eq. 5, 6 and 7, and $\mathcal{B}_{2mk(j+1)}$'s simulation for the challenge ciphertext is the same as that expressed in Eq. 8, except that except that $(\gamma_{\mathsf{dec}}^{(1)}\overrightarrow{e}_1^{(1)} + \sigma_{\mathsf{dec},1}^{(1)}\overrightarrow{x}_1, \ldots, \sigma_{\mathsf{dec},l}^{(1)}\overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)} + \overrightarrow{v}_1'$ and $(\gamma_{\mathsf{dec}}^{(2)}\overrightarrow{e}_1^{(2)} + \sigma_{\mathsf{dec},1}^{(2)}\overrightarrow{I}_1, \ldots, \sigma_{\mathsf{dec},r}^{(2)}\overrightarrow{I}_r, 0^{L-2r}) \cdot Z^{(2)} + \overrightarrow{v}_2'$ where $\overrightarrow{v}_1' \xleftarrow{\cup} \mathbb{F}_q^n, \overrightarrow{v}_2' \xleftarrow{\cup} \mathbb{F}_q^L$.

Therefore, we will show that $(\gamma_{\mathsf{dec}}^{(1)}\overrightarrow{e}_1^{(1)} + \sigma_{\mathsf{dec},1}^{(1)}\overrightarrow{x}_1, \ldots, \sigma_{\mathsf{dec},l}^{(1)}\overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)} + \overrightarrow{v}_1'$ and $(\gamma_{\mathsf{dec}}^{(2)}\overrightarrow{e}_1^{(2)} + \sigma_{\mathsf{dec},1}^{(2)}\overrightarrow{I}_1, \ldots, \sigma_{\mathsf{dec},r}^{(2)}\overrightarrow{I}_r, 0^{L-2r}) \cdot Z^{(2)} + \overrightarrow{v}_2'$ are uniformly distributed and independent from the other variables used in the simulation by $\mathcal{B}_{2mk(j+1)}$. Let $\overrightarrow{A}_1 = (\varrho_1\overrightarrow{e}_1^{(1)} + \theta_1'\overrightarrow{x}_1, \ldots, \theta_l'\overrightarrow{x}_l, 0^{n-\mu_l}) \cdot Z^{(1)} + \overrightarrow{v}_1'$, $\overrightarrow{A}_2 = (\varrho_2\overrightarrow{e}_1^{(2)} + \theta_1''\overrightarrow{I}_1, \ldots, \theta_\kappa''\overrightarrow{I}_\kappa) \cdot$

$Z^{(2)}+\overrightarrow{v}'_2$, and $\overrightarrow{B}_1=(\overrightarrow{y}^{(b)}_1,\ldots,\overrightarrow{y}^{(b)}_{h^{(b)}},\overrightarrow{y}_{h^{(b)}+1},\ldots,\overrightarrow{y}_d)\cdot U^{(1)}$, $\overrightarrow{B}_2=(\overrightarrow{I}'_1,\ldots,\overrightarrow{I}'_\kappa)\cdot U^{(2)}$ where $\overrightarrow{I}_z=(1,-i_z)$ and $i_1,\ldots,i_\kappa$ is parsed from I. We analyze joint distribution of these variables for the cases that appear in **Definition 4**.

1. If $i>I$, i.e., the time period of the queried key is after the time period of encoded in the ciphertext, then due to Lemma 3, the pair $(\overrightarrow{A}_2,\overrightarrow{B}_2)$ is uniformly and independently distributed over $\mathbb{F}_q^{2n}$.
2. If $i\leq I$ and $f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}^{(0)}_1,\ldots,\overrightarrow{y}^{(0)}_{h^{(0)}})=f_{(\overrightarrow{x}_1,\ldots,\overrightarrow{x}_l)}(\overrightarrow{y}^{(1)}_1,\ldots,\overrightarrow{y}^{(1)}_{h^{(1)}})=0$ then the pair $(\overrightarrow{A}_2,\overrightarrow{B}_2)$ is uniformly and independently distributed over $\mathbb{F}_q^{2n}$ and $(\overrightarrow{A}_1,\overrightarrow{B}_1)$ is uniformly and independently distributed over $\mathbb{F}_q^{2L}$.

Considering the adversary $\mathcal{A}$'s restriction on key queries from **Definition 4**, in above two cases at least one of $(\overrightarrow{A}_1,\overrightarrow{B}_1)$ and $(\overrightarrow{A}_2,\overrightarrow{B}_2)$ is uniformly and independently distributed over $\mathbb{F}_q^{2n_k}$ for $k=1,2$. Therefore, $(\gamma^{(1)}_{\mathsf{dec}}\overrightarrow{e}^{(1)}_1+\sigma^{(1)}_{\mathsf{dec},1}\overrightarrow{x}_1,\ldots,\sigma^{(1)}_{\mathsf{dec},l}\overrightarrow{x}_l,0^{n-\mu_l})\cdot Z^{(1)}+\overrightarrow{v}'_1$ and $(\gamma^{(2)}_{\mathsf{dec}}\overrightarrow{e}^{(2)}_1+\sigma^{(2)}_{\mathsf{dec},1}\overrightarrow{I}_1,\ldots,\sigma^{(2)}_{\mathsf{dec},r}\overrightarrow{I}_r,0^{L-2r})\cdot Z^{(2)}+\overrightarrow{v}'_2$ are uniformly distributed and independent from the other variables used in the simulation by $\mathcal{B}_{2mk(j+1)}$.

Therefore, the view of $\mathcal{A}$ in the game simulated by $\mathcal{B}_{2mk(j+1)}$ on input an instance of Problem 2 with $\beta=1$ is the same as in Game 2-(m,k,j') unless $\omega=0$ occurs. This event happens with probability $1/q$.

This completes the proof of **Lemma 6**. □

Lemma 7. *For any adversary $\mathcal{A}$, $\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda)\leq\mathsf{Adv}^{(2\text{-}(\nu,0,0))}_{\mathcal{A}}(\lambda)+1/q$.*

Proof. First we show the distribution $(\mathsf{param}_{\overrightarrow{n}},\{\widehat{\mathbb{B}}^{(k)}\}_{k=0,1,2},\{SK^{(j)}_{i,l}\}_{j=1,\ldots,\nu},C=(\boldsymbol{c}^{(0)},\boldsymbol{c}^{(1)},\boldsymbol{c}^{(2)},\boldsymbol{c}^{(M)}))$ of Game 3 is same as that of Game 2-$(\nu,0,0)$, where $SK^{(j)}_{i,l}$ is the answer to the j-th key query, and $C=(\boldsymbol{c}^{(0)},\boldsymbol{c}^{(1)},\boldsymbol{c}^{(2)},\boldsymbol{c}^{(M)})$ is the challenge ciphertext. We will define new bases $\mathbb{D}^{(k)}$ of $\mathbb{V}_k$ and $\mathbb{D}^{*(k)}$ of $\mathbb{V}^*_k$ for $k=0,1,2$.

For $k=0$, we set $\boldsymbol{d}^{(0)}_2=\boldsymbol{b}^{(0)}_2-\lambda\boldsymbol{b}^{(0)}_3$ and $\boldsymbol{d}^{*(0)}_3=\boldsymbol{b}^{*(0)}_3+\lambda\boldsymbol{b}^{*(0)}_2$, where $\lambda\xleftarrow{\mathsf{U}}\mathbb{F}_q$. The new bases are $\mathbb{D}^{(0)}=(\boldsymbol{b}^{(0)}_1,\boldsymbol{d}^{(0)}_2,\boldsymbol{b}^{(0)}_3,\boldsymbol{b}^{(0)}_4,\boldsymbol{b}^{(0)}_5)$ and $\mathbb{D}^{*(0)}=(\boldsymbol{b}^{*(0)}_1,\boldsymbol{b}^{*(0)}_2,\boldsymbol{d}^{*(0)}_3,\boldsymbol{b}^{*(0)}_4,\boldsymbol{b}^{*(0)}_5)$. We can easily verify that $\mathbb{D}^{(0)}$ and $\mathbb{D}^{*(0)}$ are dual orthonormal, and are distributed the same as the original bases $\mathbb{B}^{(0)}$ and $\mathbb{B}^{*(0)}$ respectively.

For $i,j=1,\ldots,n$, choose $Q^{(1)}=(\mu^{(1)}_{i,j})\xleftarrow{\mathsf{U}}\mathbb{F}_q^{n\times n}$, and compute $\boldsymbol{d}^{(1)}_{n+i}=\boldsymbol{b}^{(1)}_{n+i}+\sum_{j=1}^n\mu^{(1)}_{i,j}\boldsymbol{b}^{(1)}_j$, $\boldsymbol{d}^{*(1)}_i=\boldsymbol{b}^{*(1)}_i-\sum_{j=1}^n\mu^{(1)}_{j,i}\boldsymbol{b}^{*(1)}_{n+j}$, which are equivalent to the following matrix computations:

$$\begin{pmatrix}\overrightarrow{B}^{(1)}_1\\ \overrightarrow{D}^{(1)}_2\end{pmatrix}=\begin{pmatrix}I_n & 0_n\\ Q^{(1)} & I_n\end{pmatrix}\begin{pmatrix}\overrightarrow{B}^{(1)}_1\\ \overrightarrow{B}^{(1)}_2\end{pmatrix},\qquad\begin{pmatrix}\overrightarrow{D}^{*(1)}_1\\ \overrightarrow{B}^{*(1)}_2\end{pmatrix}=\begin{pmatrix}I_n & -Q^{\mathrm{T}(1)}\\ 0_n & I_n\end{pmatrix}\begin{pmatrix}\overrightarrow{B}^{*(1)}_1\\ \overrightarrow{B}^{*(1)}_2\end{pmatrix}.$$

where $\overrightarrow{B}^{(1)}_1=(\boldsymbol{b}^{(1)}_1,\ldots,\boldsymbol{b}^{(1)}_n)^{\mathrm{T}}$, $\overrightarrow{B}^{(1)}_2=(\boldsymbol{b}^{(1)}_{n+1},\ldots,\boldsymbol{b}^{(1)}_{2n})^{\mathrm{T}}$, $\overrightarrow{B}^{*(1)}_1=(\boldsymbol{b}^{*(1)}_1,\ldots,\boldsymbol{b}^{*(1)}_n)^{\mathrm{T}}$, $\overrightarrow{B}^{*(1)}_2=(\boldsymbol{b}^{*(1)}_{n+1},\ldots,\boldsymbol{b}^{*(1)}_{2n})^{\mathrm{T}}$, $\overrightarrow{D}^{(1)}_2=(\boldsymbol{d}^{(1)}_{n+1},\ldots,\boldsymbol{d}^{(1)}_{2n})^{\mathrm{T}}$, $\overrightarrow{D}^{*(1)}_1=(\boldsymbol{d}^{*(1)}_1,\ldots,\boldsymbol{d}^{*(1)}_n)^{\mathrm{T}}$.

The new bases are $\mathbb{D}^{(1)}=(\boldsymbol{b}^{(1)}_1,\ldots,\boldsymbol{b}^{(1)}_n,\boldsymbol{d}^{(1)}_{n+1},\ldots,\boldsymbol{d}^{(1)}_{2n},\boldsymbol{b}^{(1)}_{2n+1},\ldots,\boldsymbol{b}^{(1)}_{3n+1})$ and $\mathbb{D}^{*(1)}=(\boldsymbol{d}^{*(1)}_1,\ldots,\boldsymbol{d}^{*(1)}_n,\boldsymbol{b}^{*(1)}_{n+1},\ldots,\boldsymbol{b}^{*(1)}_{2n},\boldsymbol{b}^{*(1)}_{2n+1},\ldots,\boldsymbol{b}^{*(1)}_{3n+1})$. It is clear that $\mathbb{D}^{(1)}$ and $\mathbb{D}^{*(1)}$ are dual orthonormal and have the same distribution as the original bases $\mathbb{B}^{(1)}$ and $\mathbb{B}^{*(1)}$ respectively. Similarly, distribution of bases $\mathbb{D}^{(2)}$ and $\mathbb{D}^{*(2)}$ is the same as of $\mathbb{B}^{(2)}$ and $\mathbb{B}^{*(2)}$, respectively.

Keys and challenge ciphertext $(\{SK^{(j)}_{i,l}\}_{j=1,\ldots,\nu},C=(\boldsymbol{c}^{(0)},\boldsymbol{c}^{(1)},\boldsymbol{c}^{(2)},\boldsymbol{c}^{(M)}))$ of Game 2-ν are expressed over bases $\mathbb{B}^{(k)}$ and $\mathbb{B}^{*(k)}$ for $k=0,1,2$ as follows:

Parse $SK^{(j)}_{i,l}$ as $(sk^{(j)}_{i,l},\{sk^{(j)}_{i|_{z-1}1,l}\}_{i_z=0})$. Parse each $sk_{w,l}$ as $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec},j},\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec},j},\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec},j},\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},1,j},\ldots,\boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},l+1,j},\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},1,j},\ldots,\boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},r+1,j},\boldsymbol{k}^{(1)}_{w,l,\mathsf{del},\mu_l+1,j},\ldots,\boldsymbol{k}^{(1)}_{w,l,\mathsf{del},n,j},\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},(2r+1),j},\ldots,\boldsymbol{k}^{(2)}_{w,l,\mathsf{del},L,j})$.
For simplicity we only show the transformation of decryption key component $(\boldsymbol{k}^{(0)}_{w,l,\mathsf{dec},j},\boldsymbol{k}^{(1)}_{w,l,\mathsf{dec},j},\boldsymbol{k}^{(2)}_{w,l,\mathsf{dec},j})$, The randomness and the delegation components can be constructed in a similar way.

$$\boldsymbol{k}^{(0)}_{i,l,\mathsf{dec},j}=(-\alpha_j,\epsilon_j,1,\eta_j,0)_{\mathbb{B}^{*(0)}},$$

$$\boldsymbol{k}^{(1)}_{i,l,\mathsf{dec},j} = (\overbrace{\alpha^{(1)}_j \overrightarrow{e}^{(1)}_1 + \beta^{(1)}_{\mathsf{dec},1,j} \overrightarrow{x}_{1,j}, \ldots, \beta^{(1)}_{\mathsf{dec},l,j} \overrightarrow{x}_{l,j}, 0^{n-\mu_l}}^{n}, \overbrace{\gamma^{(1)}_{1,j}, \ldots, \gamma^{(1)}_{n,j}}^{n}, \overbrace{\eta^{(1)}_{1,j}, \ldots, \eta^{(1)}_{n,j}}^{n}, \overbrace{0}^{1})_{\mathbb{B}^{*(1)}},$$
$$\boldsymbol{k}^{(2)}_{i,l,\mathsf{dec},j} = (\overbrace{\alpha^{(2)}_j \overrightarrow{e}^{(2)}_1 + \beta^{(2)}_{\mathsf{dec},1,j} \overrightarrow{I}_{1,j}, \ldots, \beta^{(2)}_{\mathsf{dec},\kappa,j} \overrightarrow{I}_{\kappa,j}}^{L}, \overbrace{\gamma^{(2)}_{1,j}, \ldots, \gamma^{(2)}_{L,j}}^{L}, \overbrace{\eta^{(2)}_{1,j}, \ldots, \eta^{(2)}_{L,j}}^{L}, \overbrace{0}^{1})_{\mathbb{B}^{*(2)}}.$$

$$\begin{aligned}
\boldsymbol{c}^{(0)} &= (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}}, \\
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}^{(b)}_1, \ldots, \overrightarrow{y}^{(b)}_{h^{(b)}}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d), \overrightarrow{w}_1, 0^n, \varphi^{(1)})_{\mathbb{B}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1,-i_1), \ldots, (1,-i_\kappa)), \overrightarrow{w}_2, 0^L, \varphi^{(2)})_{\mathbb{B}^{(2)}}, \\
\boldsymbol{c}^{(M)} &= g_T^{\zeta} M^{(b)}.
\end{aligned}$$

Above keys and challenge ciphertext can also be expressed over bases $\mathbb{D}^{(k)}$ and $\mathbb{D}^{*(k)}$ for $k = 0, 1, 2$ as:

$$\boldsymbol{k}^{(0)}_{i,l,\mathsf{dec},j} = (-\alpha_j, \epsilon_j, 1, \eta_j, 0)_{\mathbb{B}^{*(0)}} = (-\alpha_j, \theta_j, 1, \eta_j, 0)_{\mathbb{D}^{*(0)}},$$

where $\theta_j = \epsilon_j - \lambda$ which are uniformly, independently distributed since $\epsilon_j \xleftarrow{\cup} \mathbb{F}_q$.

$$\begin{aligned}
\boldsymbol{k}^{(1)}_{i,l,\mathsf{dec},j} &= (\overbrace{\alpha^{(1)}_j \overrightarrow{e}^{(1)}_1 + \beta^{(1)}_{\mathsf{dec},1,j} \overrightarrow{x}_{1,j}, \ldots, \beta^{(1)}_{\mathsf{dec},l,j} \overrightarrow{x}_{l,j}, 0^{n-\mu_l}}^{n}, \overbrace{\gamma^{(1)}_{1,j}, \ldots, \gamma^{(1)}_{n,j}}^{n}, \overbrace{\eta^{(1)}_{1,j}, \ldots, \eta^{(1)}_{n,j}}^{n}, \overbrace{0}^{1})_{\mathbb{B}^{*(1)}} \\
&= (\overbrace{\alpha^{(1)}_j \overrightarrow{e}^{(1)}_1 + \beta^{(1)}_{\mathsf{dec},1,j} \overrightarrow{x}_{1,j}, \ldots, \beta^{(1)}_{\mathsf{dec},l,j} \overrightarrow{x}_{l,j}, 0^{n-\mu_l}}^{n}, \overbrace{\theta^{(1)}_{1,j}, \ldots, \theta^{(1)}_{n,j}}^{n}, \overbrace{\eta^{(1)}_{1,j}, \ldots, \eta^{(1)}_{n,j}}^{n}, \overbrace{0}^{1})_{\mathbb{D}^{*(1)}} \\
\boldsymbol{k}^{(2)}_{i,l,\mathsf{dec},j} &= (\overbrace{\alpha^{(2)}_j \overrightarrow{e}^{(2)}_1 + \beta^{(2)}_{\mathsf{dec},1,j} \overrightarrow{I}_{1,j}, \ldots, \beta^{(2)}_{\mathsf{dec},\kappa,j} \overrightarrow{I}_{\kappa,j}}^{L}, \overbrace{\gamma^{(2)}_{1,j}, \ldots, \gamma^{(2)}_{L,j}}^{L}, \overbrace{\eta^{(2)}_{1,j}, \ldots, \eta^{(2)}_{L,j}}^{L}, \overbrace{0}^{1})_{\mathbb{B}^{*(2)}} \\
&= (\overbrace{\alpha^{(2)}_j \overrightarrow{e}^{(2)}_1 + \beta^{(2)}_{\mathsf{dec},1,j} \overrightarrow{I}_{1,j}, \ldots, \beta^{(2)}_{\mathsf{dec},\kappa,j} \overrightarrow{I}_{\kappa,j}}^{L}, \overbrace{\gamma^{(2)}_{1,j}, \ldots, \gamma^{(2)}_{L,j}}^{L}, \overbrace{\eta^{(2)}_{1,j}, \ldots, \eta^{(2)}_{L,j}}^{L}, \overbrace{0}^{1})_{\mathbb{D}^{*(2)}},
\end{aligned}$$

where $\theta^{(1)}_{i,j} = \mu^{(1)}_{i,1} \alpha^{(1)}_j + \beta^{(1)}_{\mathsf{dec},1,j} \overrightarrow{x}_{1,j} \cdot \overrightarrow{\mu}^{(1)}_{i_1} + \cdots + \beta^{(1)}_{\mathsf{dec},l,j} \overrightarrow{x}_{l,j} \cdot \overrightarrow{\mu}^{(1)}_{i_l} + \gamma^{(1)}_{i,j}$ for $i = 1, \ldots, n$, $j = 1, \ldots, \nu$, which are uniformly, independently distributed since $\gamma^{(1)}_{i,j} \xleftarrow{\cup} \mathbb{F}_q$.

$$\boldsymbol{c}_0 = (\delta, w, \zeta, 0, \varphi)_{\mathbb{B}^{(0)}} = (\delta, w, \zeta', 0, \varphi)_{\mathbb{D}^{(0)}},$$

where $\zeta' = \zeta + \lambda w$ which are uniformly, independently distributed since $w, \zeta \xleftarrow{\cup} \mathbb{F}_q$.

$$\begin{aligned}
\boldsymbol{c}^{(1)} &= (\delta(\overrightarrow{y}^{(b)}_1, \ldots, \overrightarrow{y}^{(b)}_{h^{(b)}}, \overrightarrow{y}_{h^{(b)}+1}, \ldots, \overrightarrow{y}_d), \overrightarrow{w}_1, 0^n, \varphi^{(1)})_{\mathbb{B}^{(1)}} \\
&= (\delta(\overrightarrow{y}'_1, \ldots, \overrightarrow{y}'_d), \overrightarrow{w}_1, 0^n, \varphi^{(1)})_{\mathbb{D}^{(1)}}, \\
\boldsymbol{c}^{(2)} &= (\delta((1,-i_1), \ldots, (1,-i_\kappa)), \overrightarrow{w}_2, 0^L, \varphi^{(2)})_{\mathbb{B}^{(2)}} \\
&= (\delta((1,-i_1), \ldots, (1,-i_\kappa)), \overrightarrow{w}_2, 0^L, \varphi^{(2)})_{\mathbb{D}^{(2)}},
\end{aligned}$$

where $y'_i = \delta y^{(b)}_i - \overrightarrow{w}_1 \overrightarrow{\mu}^{(1)}_i$ for $i = 1, \ldots, n$, which is uniformly and independently distributed since $\overrightarrow{w}_1 \xleftarrow{\cup} \mathbb{F}_q^n$. In the light of the adversary's view, both $(\mathbb{B}^{(k)}, \mathbb{B}^{*(k)})$ and $(\mathbb{D}^{(k)}, \mathbb{D}^{*(k)})$ for $k = 0, 1, 2$ are consistent with public key $(1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\widehat{\mathbb{B}}^{(k)}\}_{k=0,1,2})$. Therefore, $\{SK^{(j)}_{i,l}\}_{j=1,\ldots,\nu}$ and C can be expressed as keys and ciphertext in two ways, in Game 2-$(\nu, 0, 0)$ over bases $(\mathbb{B}^{(k)}, \mathbb{B}^{*(k)})$ and in Game 3 over bases $(\mathbb{D}^{(k)}, \mathbb{D}^{*(k)})$. Thus, Game 2-$(\nu, 0, 0)$ can be conceptually changed to Game 3 if $w \neq 0$ (in $\boldsymbol{c}^{(0)}$), i.e., except with probability $1/q$. □

Lemma 8. *For any adversary $\mathcal{A}$, $\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) = 0$.*

Proof. The value of b is independent from the adversarys view in Game 3. Hence, $\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) = 0$. □

B.1 Proof of Lemmas 1 and 2

In order to reduce the DLIN problem to Problems 1 and 2 from Definitions 6 and 7, respectively, we further introduce three "basic problems" that will serve in intermediate steps of the reduction:

- Basic Problem 0 in Definition 8.
- Basic Problem 1 in Definition 9.
- Basic Problem 2 in Definition 10.

In order to prove **Lemmas** 1 and 2 we use two intermediate **Lemmas** 9 and 10 which are two common lemmas in the proofs of **Lemmas** 1 and 2:

Lemma 9. *Let $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ be dual pairing vector spaces by direct product of symmetric pairing groups. Using $\{\phi_{i,j}\}$, we can efficiently sample a random linear transformation $W = \sum_{i=1,j=1}^{N,N} r_{i,j}\phi_{i,j}$ of $\mathbb{V}$ with random coefficients $\{r_{i,j}\}_{i,j\in\{1,\ldots,N\}} \xleftarrow{\mathsf{U}} GL(N, \mathbb{F}_q)$. The matrix $(r^*_{i,j}) = (\{r_{i,j}\}^{-1})^T$ defines the adjoint action on $\mathbb{V}$ for pairing e, i.e., $e(W(\boldsymbol{x}), (W^{-1})^T(\boldsymbol{y})) = e(\boldsymbol{x}, \boldsymbol{y})$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{V}$, where $(W^{-1})^T = \sum_{i=1,j=1}^{N,N} r^*_{i,j}\phi_{i,j}$.*

Definition 8 (Basic Problem 0). *Basic Problem 0 is to decide bit β, given $(\mathsf{param}_{\mathrm{BP0}}, \widehat{\mathbb{B}}, \mathbb{B}^*, \boldsymbol{y}^*_\beta, \boldsymbol{f}, bG, aG, acG) \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathrm{BP0}}_\beta(1^\lambda)$ for $\beta \xleftarrow{\mathsf{U}} \{0,1\}$ with probability non-negligibly better than by a random guess, where*

$$
\begin{aligned}
&\mathcal{G}^{\mathrm{BP0}}_\beta\ (1^\lambda):\\
&\quad \mathsf{param}_{\mathbb{G}} = (q, \mathbb{G}, \mathbb{G}_T, G, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bpg}}(1^\lambda),\\
&\quad \mathsf{param}_{\mathbb{V}} = (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, 3, \mathsf{param}_{\mathbb{G}}),\\
&\quad \Lambda = (\lambda_{i,j}) \xleftarrow{\mathsf{U}} GL(3, \mathbb{F}_q),\ (\mu_{i,j}) = (\Lambda^T)^{-1},\ \ b, a \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times,\\
&\quad \boldsymbol{b}_i = b\sum_{j=1}^{3} \lambda_{i,j}\boldsymbol{a}_j,\ i = 1, 3,\ \widehat{\mathbb{B}} = (\boldsymbol{b}_1, \boldsymbol{b}_3),\\
&\quad \boldsymbol{b}^*_i = a\sum_{j=1}^{3} \mu_{i,j}\boldsymbol{a}_j,\ i = 1, 2, 3,\ \mathbb{B}^* = (\boldsymbol{b}^*_1, \boldsymbol{b}^*_2, \boldsymbol{b}^*_3),\\
&\quad g_T = e(G, G)^{ab},\ \ \mathsf{param}_{\mathrm{BP0}} = (\mathsf{param}_{\mathbb{V}}, g_T),\\
&\quad \delta, \sigma, \omega \xleftarrow{\mathsf{U}} \mathbb{F}_q,\ \ \rho, \tau \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times,\\
&\quad \boldsymbol{y}^*_0 = (\delta, 0, \sigma)_{\mathbb{B}^*},\ \ \boldsymbol{y}^*_1 = (\delta, \rho, \sigma)_{\mathbb{B}^*},\ \ \boldsymbol{f} = (\omega, \tau, 0)_{\mathbb{B}},\\
&\quad \mathrm{Output}\ (\mathsf{param}_{\mathrm{BP0}}, \widehat{\mathbb{B}}, \mathbb{B}^*, \boldsymbol{y}^*_\beta, \boldsymbol{f}, bG, aG, acG).
\end{aligned}
$$

Let $\mathsf{Adv}^{\mathrm{BP0}}_{\mathcal{F}}(\lambda)$ denote the corresponding advantage of a PPT algorithm $\mathcal{F}$ for the Basic Problem 0.

Lemma 10. *For any adversary $\mathcal{F}$, there exists a probabilistic machine $\mathcal{D}$, whose running time is essentially the same as that of $\mathcal{D}$, such that for any security parameter λ, $\mathsf{Adv}^{\mathrm{BP0}}_{\mathcal{F}}(\lambda) \leq \mathsf{Adv}^{\mathrm{DLIN}}_{\mathcal{D}}(\lambda) + 5/q$.*

The proof of Lemma 10 can be found in [28].

Proof of Lemma 1: Combining **Lemma** 9, 10, 11 and 12, we obtain **Lemma** 1.

Definition 9 (Basic Problem 1). *Basic Problem 1 is to decide bit β, given $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(k)}, \widehat{\mathbb{B}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{f}^{(0)}_\beta, \boldsymbol{f}^{(1)}_{\beta,1}, \boldsymbol{f}^{(2)}_{\beta,1}, \{\boldsymbol{f}^{(1)}_i\}_{i=2,\ldots,n_1}, \{\boldsymbol{f}^{(2)}_i\}_{i=2,\ldots,n_2},) \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathrm{BP1}}_\beta(1^\lambda, \overrightarrow{n} = (2; n_1, n_2))$ for $\beta \xleftarrow{\mathsf{U}} \{0,1\}$, with probability non-negligibly better than by a random guess, where*

$$\mathcal{G}^{\mathrm{BP1}}_\beta\ (1^\lambda, \overrightarrow{n} = (2; n_1, n_2)):$$

$$
\begin{aligned}
&(\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{*(2)}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}),\\
&\widehat{\mathbb{B}}^{*(0)} = (\boldsymbol{b}_1^{*(0)}, \boldsymbol{b}_3^{*(0)}, \boldsymbol{b}_4^{*(0)}, \boldsymbol{b}_5^{*(0)}),\\
&\widehat{\mathbb{B}}^{*(1)} = (\boldsymbol{b}_1^{*(1)}, \ldots, \boldsymbol{b}_{n_1}^{*(1)}, \boldsymbol{b}_{n_1+2}^{*(1)}, \ldots, \boldsymbol{b}_{3n_1+1}^{*(1)}),\\
&\widehat{\mathbb{B}}^{*(2)} = (\boldsymbol{b}_1^{*(2)}, \ldots, \boldsymbol{b}_{n_2}^{*(2)}, \boldsymbol{b}_{n_2+2}^{*(2)}, \ldots, \boldsymbol{b}_{3n_2+1}^{*(2)}),\\
&\omega, \gamma \xleftarrow{\mathsf{U}} \mathbb{F}_q, \ \ \tau \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times, \ \ \boldsymbol{f}_0^{(0)} = (\omega, 0, 0, 0, \gamma)_{\mathbb{B}^{(0)}}, \ \ \boldsymbol{f}_1^{(0)} = (\omega, \tau, 0, 0, \gamma)_{\mathbb{B}^{(0)}},\\
&\boldsymbol{f}_{0,1}^{(1)} = (\overbrace{\omega \overrightarrow{e}_1^{(1)}}^{n_1}, \overbrace{0^{n_1}}^{n_1}, \overbrace{0^{n_1}}^{n_1}, \overbrace{\gamma}^{1})_{\mathbb{B}^{(1)}},\\
&\boldsymbol{f}_{1,1}^{(1)} = (\overbrace{\omega \overrightarrow{e}_1^{(1)}}^{n_1}, \overbrace{\tau \overrightarrow{e}_1^{(1)}}^{n_1}, \overbrace{0^{n_1}}^{n_1}, \overbrace{\gamma}^{1})_{\mathbb{B}^{(1)}},\\
&\text{For } i = 2, \ldots, n_1 : \boldsymbol{f}_i^{(1)} = \omega \boldsymbol{b}_i^{(1)};\\
&\boldsymbol{f}_{0,1}^{(2)} = (\overbrace{\omega \overrightarrow{e}_1^{(2)}}^{n_2}, \overbrace{0^{n_2}}^{n_2}, \overbrace{0^{n_2}}^{n_2}, \overbrace{\gamma}^{1})_{\mathbb{B}^{(2)}},\\
&\boldsymbol{f}_{1,1}^{(2)} = (\overbrace{\omega \overrightarrow{e}_1^{(2)}}^{n_2}, \overbrace{\tau \overrightarrow{e}_1^{(2)}}^{n_2}, \overbrace{0^{n_2}}^{n_2}, \overbrace{\gamma}^{1})_{\mathbb{B}^{(2)}},\\
&\text{For } i = 2, \ldots, n_2 : \boldsymbol{f}_i^{(2)} = \omega \boldsymbol{b}_i^{(2)};\\
&\text{Output } (\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(k)}, \widehat{\mathbb{B}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{f}_\beta^{(0)}, \boldsymbol{f}_{\beta,1}^{(1)}, \boldsymbol{f}_{\beta,1}^{(2)}, \{\boldsymbol{f}_i^{(1)}\}_{i=2,\ldots,n_1}, \{\boldsymbol{f}_i^{(2)}\}_{i=2,\ldots,n_2},).
\end{aligned}
$$

Let $\mathsf{Adv}_{\mathcal{C}}^{\mathrm{BP1}}(\lambda)$ *denote the advantage of a PPT algorithm* $\mathcal{C}$ *for the Basic Problem* 1.

Lemma 11. *For any adversary* $\mathcal{C}$*, there exists a probabilistic machine* $\mathcal{F}$*, whose running time is essentially the same as that of* $\mathcal{C}$*, such that for any security parameter* λ*,* $\mathsf{Adv}_{\mathcal{C}}^{\mathrm{BP1}}(\lambda) \leq \mathsf{Adv}_{\mathcal{F}}^{\mathrm{BP0}}(\lambda)$ *for* $\overrightarrow{n} = (2; n_1, n_2)$.

Proof. $\mathcal{F}$ is given a Basic Problem 0 instance $(\mathsf{param}_{\mathrm{BP0}}, \widehat{\mathbb{B}}, \mathbb{B}^*, \boldsymbol{y}_\beta^*, \boldsymbol{f}, bG, aG, acG)$. Using $\mathsf{param}_{\mathbb{G}} = (q, \mathbb{G}, \mathbb{G}_T, G, e)$ contained in $\mathsf{param}_{\mathrm{BP0}}$, $\mathcal{F}$ computes:

$$
\begin{aligned}
&\mathsf{param}_{\mathbb{V}_0} = (q, \mathbb{V}_0, \mathbb{G}_T, \mathbb{A}_0, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, 5, \mathsf{param}_{\mathbb{G}}),\\
&\mathsf{param}_{\mathbb{V}_l} = (q, \mathbb{V}_l, \mathbb{G}_T, \mathbb{A}_l, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, 3n_l + 1, \mathsf{param}_{\mathbb{G}}), \ l = 1, 2,\\
&\mathsf{param}_{\overrightarrow{n}} = (\{\mathsf{param}_{\mathbb{V}_l}\}_{l=0,1,2}, g_T),
\end{aligned}
$$

where g_T is contained in $\mathsf{param}_{\mathrm{BP0}}$. $\mathcal{F}$ generates random linear transformation W_l on $\mathbb{V}_l (l = 0, 1, 2)$ given in Lemma 9, then sets

$$
\begin{aligned}
&\boldsymbol{d}_l^{(0)} = W_0(\boldsymbol{b}_l^*, 0, 0), \ \ l = 1, 2; \quad \boldsymbol{d}_3^{(0)} = W_0(0, 0, 0, 0, aG),\\
&\boldsymbol{d}_4^{(0)} = W_0(0, 0, 0, aG, 0), \quad \boldsymbol{d}_5^{(0)} = W_0(\boldsymbol{b}_3^*, 0, 0),\\
&\boldsymbol{d}_l^{*(0)} = (W_0^{-1})^T(\boldsymbol{b}_l, 0, 0), \ \ l = 1, 2; \quad \boldsymbol{d}_3^{*(0)} = (W_0^{-1})^T(0, 0, 0, 0, bG),\\
&\boldsymbol{d}_4^{*(0)} = (W_0^{-1})^T(0, 0, 0, bG, 0), \quad \boldsymbol{d}_5^{*(0)} = (W_0^{-1})^T(\boldsymbol{b}_3, 0, 0),\\
&\boldsymbol{g}_\beta^{(0)} = W_0(\boldsymbol{y}_\beta^*, 0, 0),
\end{aligned}
$$

$$
\begin{aligned}
&\boldsymbol{d}_1^{(1)} = W_1(\boldsymbol{b}_1^*, 0^{N_1-3}), \quad \boldsymbol{d}_{n_1+1}^{(1)} = W_1(\boldsymbol{b}_2^*, 0^{N_1-3}), \quad \boldsymbol{d}_{N_1}^{(1)} = W_1(\boldsymbol{b}_3^*, 0^{N_1-3}),\\
&\boldsymbol{d}_l^{(1)} = W_1(0^m, aG, 0^{N_1-m-1}) \text{ where } \begin{cases} m = l + 1 \text{ if } l \in \{2, \ldots, n_1\}, \\ m = l \text{ if } l \in \{n_1 + 2, \ldots, N_1 - 1\}, \end{cases}\\
&\boldsymbol{d}_1^{*(1)} = (W_1^{-1})^T(\boldsymbol{b}_1, 0^{N_1-3}), \quad \boldsymbol{d}_{n_1+1}^{*(1)} = (W_1^{-1})^T(\boldsymbol{b}_2, 0^{N_1-3}), \quad \boldsymbol{d}_{N_1}^{*(1)} = (W_1^{-1})^T(\boldsymbol{b}_3, 0^{N_1-3}),
\end{aligned}
$$

$$\boldsymbol{d}_l^{*(1)} = (W_1^{-1})^T(0^m, bG, 0^{N_1-m-1}) \text{ where } \begin{cases} m = l+1 \text{ if } l \in \{2,\ldots,n_1\}, \\ m = l \text{ if } l \in \{n_1+2,\ldots,N_1-1\}, \end{cases}$$
$$\boldsymbol{g}_{\beta,1}^{(1)} = W_1(\boldsymbol{y}_\beta^*, 0^{N_1-3}),$$
$$\boldsymbol{g}_l^{(1)} = W_1(0^{l+1}, acG, 0^{N_1-l-2}),\ \ l = 2,\ldots,n_1;$$

$$\boldsymbol{d}_1^{(2)} = W_2(\boldsymbol{b}_1^*, 0^{N_2-3}), \quad \boldsymbol{d}_{n_2+1}^{(2)} = W_2(\boldsymbol{b}_2^*, 0^{N_2-3}), \quad \boldsymbol{d}_{N_2}^{(2)} = W_2(\boldsymbol{b}_3^*, 0^{N_2-3}),$$
$$\boldsymbol{d}_l^{(2)} = W_2(0^m, aG, 0^{N_2-m-1}) \text{ where } \begin{cases} m = l+1 \text{ if } l \in \{2,\ldots,n_2\}, \\ m = l \text{ if } l \in \{n_2+2,\ldots,N_2-1\}, \end{cases}$$
$$\boldsymbol{d}_1^{*(2)} = (W_2^{-1})^T(\boldsymbol{b}_1, 0^{N_2-3}), \quad \boldsymbol{d}_{n_2+1}^{*(2)} = (W_2^{-1})^T(\boldsymbol{b}_2, 0^{N_2-3}), \quad \boldsymbol{d}_{N_2}^{*(2)} = (W_2^{-1})^T(\boldsymbol{b}_3, 0^{N_2-3}),$$
$$\boldsymbol{d}_l^{*(2)} = (W_2^{-1})^T(0^m, bG, 0^{N_2-m-1}) \text{ where } \begin{cases} m = l+1 \text{ if } l \in \{2,\ldots,n_2\}, \\ m = l \text{ if } l \in \{n_2+2,\ldots,N_2-1\}, \end{cases}$$
$$\boldsymbol{g}_{\beta,1}^{(2)} = W_2(\boldsymbol{y}_\beta^*, 0^{N_2-3}),$$
$$\boldsymbol{g}_l^{(2)} = W_2(0^{l+1}, acG, 0^{N_2-l-2}),\ \ l = 2,\ldots,n_2;$$

where $(\boldsymbol{v}, 0^{N_l-3}) = (G', G'', G''', 0^{N_l-3})$ for any $\boldsymbol{v} = (G', G'', G''') \in \mathbb{V} = \mathbb{G}^3$. In this way bases $\mathbb{D}^{(0)} = (\boldsymbol{d}_l^{(0)})_{l=1,\ldots,5}$ and $\mathbb{D}^{*(0)} = (\boldsymbol{d}_l^{*(0)})_{l=1,\ldots,5}$, $\mathbb{D}^{(j)} = (\boldsymbol{d}_l^{(j)})_{l=1,\ldots,3n_j+1}$ and $\mathbb{D}^{*(j)} = (\boldsymbol{d}_l^{*(j)})_{l=1,\ldots,3n_j+1}$, $j = 1,2$ are dual orthonormal bases.

Therefore, from $\widehat{\mathbb{B}} = (\boldsymbol{b}_1, \boldsymbol{b}_3)$, $\mathbb{B}^*$, bG, and aG the algorithm $\mathcal{F}$ can compute $\mathbb{D}^{(j)}$, $j = 0,1,2$; $\widehat{\mathbb{D}}^{*(0)} = (\boldsymbol{d}_1^{*(0)}, \boldsymbol{d}_3^{*(0)}, \boldsymbol{d}_4^{*(0)}, \boldsymbol{d}_5^{*(0)})$, and $\widehat{\mathbb{D}}^{*(j)} = (\boldsymbol{d}_1^{*(j)}, \ldots, \boldsymbol{d}_{n_j}^{*(j)}, \boldsymbol{d}_{n_j+2}^{*(j)}, \ldots, \boldsymbol{d}_{3n_j+1}^{*(j)})$, $j = 1,2$.

Finally, $\mathcal{F}$ hands $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{D}^{(k)}, \widehat{\mathbb{D}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{g}_\beta^{(0)}, \boldsymbol{g}_{\beta,1}^{(1)}, \boldsymbol{g}_{\beta,1}^{(2)}, \{\boldsymbol{g}_i^{(1)}\}_{i=2,\ldots,n_1}, \{\boldsymbol{g}_i^{(2)}\}_{i=2,\ldots,n_2},)$ over to $\mathcal{C}$ and, if $\mathcal{C}$ outputs its bit β' then $\mathcal{F}$ forwards this bit as its own output.

We observe that:

$$\boldsymbol{g}_0^{(0)} = (\omega', 0, 0, 0, \gamma')_{\mathbb{D}^{(0)}}, \qquad \boldsymbol{g}_1^{(0)} = (\omega', \tau', 0, 0, \gamma')_{\mathbb{D}^{(0)}},$$
$$\boldsymbol{g}_{0,1}^{(1)} = (\overbrace{\omega'\overrightarrow{e}_1^{(1)}}^{n_1}, \overbrace{0^{n_1}}^{n_1}, \overbrace{0^{n_1}}^{n_1}, \overbrace{\gamma'}^{1})_{\mathbb{D}^{(1)}}, \qquad \boldsymbol{g}_{0,1}^{(2)} = (\overbrace{\omega'\overrightarrow{e}_1^{(2)}}^{n_2}, \overbrace{0^{n_2}}^{n_2}, \overbrace{0^{n_2}}^{n_2}, \overbrace{\gamma'}^{1})_{\mathbb{D}^{(2)}},$$
$$\boldsymbol{g}_{1,1}^{(1)} = (\overbrace{\omega'\overrightarrow{e}_1^{(1)}}^{n_1}, \overbrace{\tau'\overrightarrow{e}_1^{(1)}}^{n_1}, \overbrace{0^{n_1}}^{n_1}, \overbrace{\gamma'}^{1})_{\mathbb{D}^{(1)}}, \qquad \boldsymbol{g}_{1,1}^{(2)} = (\overbrace{\omega'\overrightarrow{e}_1^{(2)}}^{n_2}, \overbrace{\tau'\overrightarrow{e}_1^{(2)}}^{n_2}, \overbrace{0^{n_2}}^{n_2}, \overbrace{\gamma'}^{1})_{\mathbb{D}^{(2)}}$$
$$\boldsymbol{g}_i^{(1)} = \omega'\boldsymbol{b}_i^{(1)},\ \ i = 2,\ldots,n_1; \qquad \boldsymbol{g}_i^{(2)} = \omega'\boldsymbol{b}_i^{(2)},\ \ i = 2,\ldots,n_2;$$

where $\omega' = \delta, \tau' = \rho, \gamma' = \sigma$ are distributed uniformly in $\mathbb{F}_q$, Therefore, the distribution of $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{D}^{(k)}, \widehat{\mathbb{D}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{g}_\beta^{(0)}, \boldsymbol{g}_{\beta,1}^{(1)}, \boldsymbol{g}_{\beta,1}^{(2)}, \{\boldsymbol{g}_i^{(1)}\}_{i=2,\ldots,n_1}, \{\boldsymbol{g}_i^{(2)}\}_{i=2,\ldots,n_2})$ is exactly the same as in the instance of Basic Problem 1. □

Lemma 12. *For any adversary $\mathcal{B}$, there exists a probabilistic machine $\mathcal{C}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}_{\mathcal{B}}^{\mathrm{P1}}(\lambda) \leq \mathsf{Adv}_{\mathcal{C}}^{\mathrm{BP1}}(\lambda) + 3/q$ for $(\overrightarrow{n} = (2; n_1, n_2))$.*

Proof. $\mathcal{C}$ is given an instance of the Basic Problem 1, that is a tuple $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}^{(k)}, \widehat{\mathbb{B}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{f}_\beta^{(0)}, \boldsymbol{f}_{\beta,1}^{(1)}, \boldsymbol{f}_{\beta,1}^{(2)}, \{\boldsymbol{f}_i^{(1)}\}_{i=2,\ldots,n_1}, \{\boldsymbol{f}_i^{(2)}\}_{i=2,\ldots,n_2})$, and computes $\boldsymbol{r} \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}_{3n_1+1}^{(1)} \rangle$, $\boldsymbol{r}' \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}_{3n_2+1}^{(2)} \rangle$, and sets $\boldsymbol{t}_{\beta,1}^{(1)} = \boldsymbol{f}_{\beta,1}^{(1)} + \boldsymbol{r}$ and $\boldsymbol{t}_{\beta,1}^{(2)} = \boldsymbol{f}_{\beta,1}^{(2)} + \boldsymbol{r}'$.

Then, $\mathcal{C}$ chooses $u_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times$, $(u_{i,j}^{(k)}) \xleftarrow{\mathsf{U}} GL(\mathbb{F}_q, n_k)$, $(z_{i,j}^{(k)}) = \left((u_{i,j}^{(k)})^{-1}\right)^T$ for $i = 1,\ldots,n_k, j = 1,\ldots,n_k$, and $k = 1,2$, and computes:

$$\boldsymbol{d}_2^{(0)} = (0, u_0, 0, 0, 0)_{\mathbb{B}^{(0)}},$$

$$\boldsymbol{d}^{(k)}_{n_k+i} = (\overbrace{0^{n_k}}^{n_k}, \overbrace{u^{(k)}_{i,1}, \ldots, u^{(k)}_{i,n_k}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{(k)}},\ i = 1, \ldots, n_k,\ k = 1, 2.$$

$\mathcal{C}$ then sets dual orthonormal basis vectors

$$\boldsymbol{d}^{*(0)}_2 = (0, u_0^{-1}, 0, 0, 0)_{\mathbb{B}^{*(0)}},$$
$$\boldsymbol{d}^{*(k)}_{n_k+i} = (\overbrace{0^{n_k}}^{n_k}, \overbrace{z^{(k)}_{i,1}, \ldots, z^{(k)}_{i,n_k}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}},\ i = 1, \ldots, n_k,\ k = 1, 2.$$

Note that $\mathcal{C}$ cannot compute $\boldsymbol{d}^{*(0)}_2$ and $\boldsymbol{d}^{*(k)}_{n_k+i}$, $i = 1, \ldots, n_k$, $k = 1, 2$ due to the lack of $\boldsymbol{b}^{*(0)}_2$ and $\boldsymbol{b}^{*(k)}_{n_k+1}$.

Then, $\mathcal{C}$ sets bases $\mathbb{D}^{(0)} = (\boldsymbol{b}^{(0)}_1, \boldsymbol{d}^{(0)}_2, \boldsymbol{b}^{(0)}_3, \boldsymbol{b}^{(0)}_4, \boldsymbol{b}^{(0)}_5)$, $\widehat{\mathbb{D}}^{*(0)} = (\boldsymbol{b}^{*(0)}_1, \boldsymbol{b}^{*(0)}_3, \boldsymbol{b}^{*(0)}_4, \boldsymbol{b}^{*(0)}_5)$, $\mathbb{D}^{(k)} = (\boldsymbol{b}^{(k)}_1, \ldots, \boldsymbol{b}^{(k)}_{n_k}, \boldsymbol{d}^{(k)}_{n_k+1}, \ldots, \boldsymbol{d}^{(k)}_{2n_k}, \boldsymbol{b}^{(k)}_{2n_k+1}, \ldots, \boldsymbol{b}^{(k)}_{3n_k+1})$, $\widehat{\mathbb{D}}^{*(k)} = (\boldsymbol{b}^{*(k)}_1, \ldots, \boldsymbol{b}^{*(k)}_{n_k}, \boldsymbol{b}^{*(k)}_{2n_k+1}, \ldots, \boldsymbol{b}^{*(k)}_{3n_k+1})$, $k = 1, 2$.

Finally, $\mathcal{C}$ hands $(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{D}^{(k)}, \widehat{\mathbb{D}}^{*(k)}\}_{k=0,1,2}, \boldsymbol{f}^{(0)}_\beta, \boldsymbol{t}^{(1)}_{\beta,1}, \boldsymbol{t}^{(2)}_{\beta,1}, \{\boldsymbol{f}^{(1)}_i\}_{i=2,\ldots,n_1}, \{\boldsymbol{f}^{(2)}_i\}_{i=2,\ldots,n_2})$ over to $\mathcal{B}$ and, if $\mathcal{B}$ outputs its bit β' then $\mathcal{C}$ forwards this bit as its own output. Note that with respect to $\mathbb{D}^{(k)}, \mathbb{D}^{*(k)}$, $k = 0, 1, 2$, the above input to $\mathcal{B}$ has the same distribution as the instance of the Problem 1 unless following events occur: $u = 0$, $\overrightarrow{u}^{(1)} = \overrightarrow{0}$, or $\overrightarrow{u}^{(2)} = \overrightarrow{0}$. Those events occur with probability $3/q$ when $\beta = 1$. □

Proof of Lemma 2: Combining **Lemmas** 9, 10, 13 and 14, we obtain **Lemma** 2.

Definition 10 (Basic Problem 2). *Basic Problem* 2 *is to find bit* β*, given* $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{y}^{*(0)}_\beta, \boldsymbol{f}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{y}^{*(k)}_{\beta,i}, \boldsymbol{f}^{(k)}_i\}_{i=1,\ldots,n_k}\}_{k=1,2}) \xleftarrow{R} \mathcal{G}^{\mathrm{BP2}}_\beta(1^\lambda, \overrightarrow{n} = (2; n_1, n_2))$ *for* $\beta \xleftarrow{\cup} \{0, 1\}$ *with probability non-negligibly better than by a random guess, where*

$\mathcal{G}^{\mathrm{BP2}}_\beta\ (1^\lambda, \overrightarrow{n} = (2; n_1, n_2)) :$

$(\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{*(2)}) \xleftarrow{R} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n})$,

$\widehat{\mathbb{B}}^{(0)} = (\boldsymbol{b}^{(0)}_1, \boldsymbol{b}^{(0)}_3, \boldsymbol{b}^{(0)}_4, \boldsymbol{b}^{(0)}_5)$,

$\widehat{\mathbb{B}}^{(1)} = (\boldsymbol{b}^{(1)}_1, \ldots, \boldsymbol{b}^{(1)}_{n_1}, \boldsymbol{b}^{(1)}_{2n_1+1}, \ldots, \boldsymbol{b}^{(1)}_{3n_1+1})$,

$\widehat{\mathbb{B}}^{(2)} = (\boldsymbol{b}^{(2)}_1, \ldots, \boldsymbol{b}^{(2)}_{n_2}, \boldsymbol{b}^{(2)}_{2n_2+1}, \ldots, \boldsymbol{b}^{(2)}_{3n_2+1})$,

$\omega, \xi, \delta \xleftarrow{\cup} \mathbb{F}_q,\ \ z, \pi \xleftarrow{\cup} \mathbb{F}_q^\times$,

$\boldsymbol{y}^{*(0)}_0 = (\omega, 0, 0, \xi, 0)_{\mathbb{B}^{*(0)}},\ \ \boldsymbol{y}^{*(0)}_1 = (\omega, z, 0, \xi, 0)_{\mathbb{B}^{*(0)}},\ \ \boldsymbol{f}^{(0)} = (\delta, \pi, 0, 0, 0)_{\mathbb{B}^{(0)}}$,

For $k = 1, 2$ and $i = 1, \ldots, n_k$:

$\boldsymbol{y}^{*(k)}_{0,i} = (\overbrace{\omega\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{\xi\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}}$,

$\boldsymbol{y}^{*(k)}_{1,i} = (\overbrace{\omega\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{z\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{\xi\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}}$,

$\boldsymbol{f}^{(k)}_i = (\overbrace{\delta\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{\pi\,\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{(k)}}$,

Output $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{y}^{*(0)}_\beta, \boldsymbol{f}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{y}^{*(k)}_{\beta,i}, \boldsymbol{f}^{(k)}_i\}_{i=1,\ldots,n_k}\}_{k=1,2})$.

Let $\mathsf{Adv}^{\mathrm{BP2}}_{\mathcal{C}}(\lambda)$ *denote the corresponding advantage of a PPT algorithm* $\mathcal{C}$ *for the Basic Problem* 2.

Lemma 13. *For any adversary* $\mathcal{C}$*, there exists a probabilistic machine* $\mathcal{F}$*, whose running time is essentially the same as that of* $\mathcal{C}$*, such that for any security parameter* λ, $\mathsf{Adv}^{\mathrm{BP2}}_{\mathcal{C}}(\lambda) = \mathsf{Adv}^{\mathrm{BP0}}_{\mathcal{F}}(\lambda)$ *for* $\overrightarrow{n} = (2; n_1, n_2)$.

Proof. $\mathcal{F}$ is given an instance of the Basic Problem 0, i.e. $(\mathsf{param}_{\mathrm{BP0}}, \widehat{\mathbb{B}}, \mathbb{B}^*, \boldsymbol{y}^*_\beta, \boldsymbol{f}, bG, aG, acG)$. Using $\mathsf{param}_{\mathbb{G}} = (q, \mathbb{G}, \mathbb{G}_T, G, e)$ contained in $\mathsf{param}_{\mathrm{BP0}}$ it computes

$$\begin{aligned}
&\mathsf{param}_{\mathbb{V}_0} = (q, \mathbb{V}_0, \mathbb{G}_T, \mathbb{A}_0, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, 5, \mathsf{param}_{\mathbb{G}}),\\
&\mathsf{param}_{\mathbb{V}_l} = (q, \mathbb{V}_l, \mathbb{G}_T, \mathbb{A}_l, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, 3n_l+1, \mathsf{param}_{\mathbb{G}}),\ l=1,2,\\
&\mathsf{param}_{\vec{n}} = (\{\mathsf{param}_{\mathbb{V}_l}\}_{l=0,1,2}, g_T),
\end{aligned}$$

where g_T is contained in $\mathsf{param}_{\mathrm{BP0}}$. Then, $\mathcal{F}$ generates random linear transformation W_l on $\mathbb{V}_l (l=0,1,2)$ given in Lemma 9 and sets

$$\begin{aligned}
&\boldsymbol{d}^{(0)}_l = W_0(\boldsymbol{b}_l, 0, 0),\ l=1,2; \quad \boldsymbol{d}^{(0)}_3 = W_0(0,0,0,0,bG),\\
&\boldsymbol{d}^{(0)}_4 = W_0(\boldsymbol{b}_3, 0, 0), \quad \boldsymbol{d}^{(0)}_5 = W_0(0,0,0,bG,0),\\
&\boldsymbol{d}^{*(0)}_l = (W_0^{-1})^T(\boldsymbol{b}^*_l, 0, 0),\ l=1,2; \quad \boldsymbol{d}^{*(0)}_3 = (W_0^{-1})^T(0,0,0,0,aG),\\
&\boldsymbol{d}^{*(0)}_4 = (W_0^{-1})^T(\boldsymbol{b}^*_3, 0, 0), \quad \boldsymbol{d}^{*(0)}_5 = (W_0^{-1})^T(0,0,0,aG,0),\\
&\boldsymbol{p}^{*(0)}_\beta = (W_0^{-1})^T(\boldsymbol{y}^*_\beta, 0, 0), \quad \boldsymbol{g}^{(0)} = W_0(\boldsymbol{f}, 0, 0),\\
&\text{For } k=1,2:\\
&\quad \text{For } l=1,2,3 \text{ and } i=1,\dots,n_k:\\
&\qquad \boldsymbol{d}^{(k)}_{(l-1)n_k+i} = W_k(0^{3(i-1)}, \boldsymbol{b}_l, 0^{3(n_k-i)}, 0),\\
&\quad \boldsymbol{d}^{(k)}_{3n_k+1} = W_k(0^{3n_k}, bG),\\
&\quad \text{For } l=1,2,3 \text{ and } i=1,\dots,n_k:\\
&\qquad \boldsymbol{d}^{*(k)}_{(l-1)n_k+i} = (W_k^{-1})^T(0^{3(i-1)}, \boldsymbol{b}^*_l, 0^{3(n_k-i)}, 0),\\
&\quad \boldsymbol{d}^{*(k)}_{3n_k+1} = (W_k^{-1})^T(0^{3n_k}, aG),\\
&\quad \text{For } i=1,\dots,n_k:\\
&\qquad \boldsymbol{p}^{*(k)}_{\beta,i} = (W_k^{-1})^T(0^{3(i-1)}, \boldsymbol{y}^*_\beta, 0^{3(n_k-i)}, 0),\\
&\qquad \boldsymbol{g}^{(k)}_i = W_1(0^{3(i-1)}, \boldsymbol{f}, 0^{3(n_k-i)}, 0).
\end{aligned}$$

Observe that $\mathbb{D}^{(0)} = (\boldsymbol{d}^{(0)}_l)_{l=1,\dots,5}$ and $\mathbb{D}^{*(0)} = (\boldsymbol{d}^{*(0)}_l)_{l=1,\dots,5}$, $\mathbb{D}^{(j)} = (\boldsymbol{d}^{(j)}_l)_{l=1,\dots,3n_j+1}$ and $\mathbb{D}^{*(j)} = (\boldsymbol{d}^{*(j)}_l)_{l=1,\dots,3n_j+1}$, $j=1,2$ are dual orthonormal bases.
Therefore, $\mathcal{F}$ can use $\widehat{\mathbb{B}} = (\boldsymbol{b}_1, \boldsymbol{b}_3)$, $\mathbb{B}^*$, bG, and aG to compute bases $\mathbb{D}^{*(j)}$, $j=0,1,2$; $\widehat{\mathbb{D}}^{(0)} = (\boldsymbol{d}^{(0)}_1, \boldsymbol{d}^{(0)}_3, \boldsymbol{d}^{(0)}_4, \boldsymbol{d}^{(0)}_5)$, and $\widehat{\mathbb{D}}^{(j)} = (\boldsymbol{d}^{(j)}_l, \dots, \boldsymbol{d}^{(j)}_{n_j}, \boldsymbol{d}^{(j)}_{2n_j+1}, \dots, \boldsymbol{d}^{(j)}_{3n_j+1})$, $j=1,2$.
Finally, $\mathcal{F}$ hands $(\mathsf{param}_{\vec{n}}, \widehat{\mathbb{D}}^{(0)}, \mathbb{D}^{*(0)}, \boldsymbol{p}^{*(0)}_\beta, \boldsymbol{g}^{(0)}, \{\widehat{\mathbb{D}}^{(k)}, \mathbb{D}^{*(k)}, \{\boldsymbol{p}^{*(k)}_{\beta,i}, \boldsymbol{g}^{(k)}_i\}_{i=1,\dots,n_k}\}_{k=1,2})$ over to $\mathcal{C}$ and, if $\mathcal{C}$ outputs a bit β', forwards this bit as its own output.
Observe that:

$$\begin{aligned}
&\boldsymbol{p}^{*(0)}_0 = (\omega, 0, 0, \xi, 0)_{\mathbb{D}^{*(0)}}, \quad \boldsymbol{p}^{*(0)}_1 = (\omega, z, 0, \xi, 0)_{\mathbb{D}^{*(0)}}, \quad \boldsymbol{g}^{(0)} = (\delta, \pi, 0, 0, 0)_{\mathbb{D}^{(0)}},\\
&\text{For } k=1,2 \text{ and } i=1,\dots,n_k:\\
&\boldsymbol{p}^{*(k)}_{0,i} = (\overbrace{\omega\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{\xi\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0}^{1})_{\mathbb{D}^{*(k)}},\\
&\boldsymbol{p}^{*(k)}_{1,i} = (\overbrace{\omega\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{z\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{\xi\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0}^{1})_{\mathbb{D}^{*(k)}},\\
&\boldsymbol{g}^{(k)}_i = (\overbrace{\delta\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{\pi\overrightarrow{e}^{(k)}_i}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{D}^{(k)}}.
\end{aligned}$$

Therefore, the distribution of $(\mathsf{param}_{\vec{n}}, \widehat{\mathbb{D}}^{(0)}, \mathbb{D}^{*(0)}, \boldsymbol{p}^{*(0)}_\beta, \boldsymbol{g}^{(0)}, \{\widehat{\mathbb{D}}^{(k)}, \mathbb{D}^{*(k)}, \{\boldsymbol{p}^{*(k)}_{\beta,i}, \boldsymbol{g}^{(k)}_i\}_{i=1,\dots,n_k}\}_{k=1,2})$ is exactly the same as in the instance of the Basic Problem 2. □

Lemma 14. *For any adversary $\mathcal{B}$, there exists a probabilistic machine $\mathcal{C}$, whose running time is essentially the same as that of $\mathcal{B}$, such that for any security parameter λ, $\mathsf{Adv}^{\mathrm{P2}}_{\mathcal{B}}(\lambda) = \mathsf{Adv}^{\mathrm{BP2}}_{\mathcal{C}}(\lambda)$.*

Proof. Given an instance of the Basic Problem 2, that is a tuple $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{B}}^{(0)}, \mathbb{B}^{*(0)}, \boldsymbol{y}^{*(0)}_{\beta}, \boldsymbol{f}^{(0)}, \{\widehat{\mathbb{B}}^{(k)}, \mathbb{B}^{*(k)}, \{\boldsymbol{y}^{*(k)}_{\beta,i}, \boldsymbol{f}^{(k)}_{i}\}_{i=1,\ldots,n_k}\}_{k=1,2})$ the algorithm $\mathcal{C}$ computes $\boldsymbol{r}^{*(k)}_i \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(k)}_{2n_k+1}, \ldots, \boldsymbol{b}^{*(k)}_{3n_k}\rangle$ and sets $\boldsymbol{h}^{*(k)}_{\beta,i} = \boldsymbol{y}^{*(k)}_{\beta,i} + \boldsymbol{r}^{*(k)}_i$, $k = 1, 2$.
Then, $\mathcal{C}$ chooses $z'_0 \xleftarrow{\mathsf{U}} \mathbb{F}^{\times}_q$, $(z'^{(k)}_{i,j}) \xleftarrow{\mathsf{U}} GL(\mathbb{F}_q, n_k), i = 1, \ldots, n_k, j = 1, \ldots, n_k, k = 1, 2$, and computes:

$$\boldsymbol{d}^{*(0)}_2 = (0, z'_0, 0, 0, 0)_{\mathbb{B}^{*(0)}},$$
$$\boldsymbol{d}^{*(k)}_{n_k+i} = (\overbrace{0^{n_k}}^{n_k}, \overbrace{z'^{(k)}_{i,1}, \ldots, z'^{(k)}_{i,n_k}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}},\ i = 1, \ldots, n_k,\ k = 1, 2,$$

Then, $\mathcal{C}$ sets $z_0 = z^{-1} z'_0, u_0 = z_0^{-1}, (z^{(k)}_{i,j}) = z^{-1}(z'^{(k)}_{i,j})$, and $(u^{(k)}_{i,j}) = \left((z^{(k)}_{i,j})^{-1}\right)^T$, where z is defined as in the Basic Problem 2. This leads to

$$\boldsymbol{d}^{*(0)}_2 = (0, z z_0, 0, 0, 0)_{\mathbb{B}^{*(0)}},$$
$$\boldsymbol{d}^{*(k)}_{n_k+i} = (\overbrace{0^{n_k}}^{n_k}, \overbrace{z z^{(k)}_{i,1}, \ldots, z z^{(k)}_{i,n_k}}^{n_k}, \overbrace{0^{n_k}}^{n_k}, \overbrace{0}^{1})_{\mathbb{B}^{*(k)}},\ i = 1, \ldots, n_k,\ k = 1, 2,$$
$$\boldsymbol{d}^{(0)}_2 = (0, z^{-1} u_0, 0, 0, 0)_{\mathbb{B}^{(0)}},$$
$$\boldsymbol{d}^{(k)}_{n_k+i} = (\overbrace{0^{n_k}}^{n_k}, \overbrace{z^{-1} u^{(k)}_{i,1}, \ldots, z^{-1} u^{(k)}_{i,n_k}}^{n_k}, \overbrace{0,}^{1} \overbrace{0}^{1})_{\mathbb{B}^{(k)}},\ i = 1, \ldots, n_k,\ k = 1, 2.$$

$\mathcal{C}$ then computes $\mathbb{D}^{*(0)} = (\boldsymbol{b}^{*(0)}_1, \boldsymbol{d}^{*(0)}_2, \boldsymbol{b}^{*(0)}_3, \boldsymbol{b}^{*(0)}_4, \boldsymbol{b}^{*(0)}_5)$, $\widehat{\mathbb{D}}^{(0)} = (\boldsymbol{b}^{(0)}_1, \boldsymbol{b}^{(0)}_3, \boldsymbol{b}^{(0)}_4, \boldsymbol{b}^{(0)}_5)$, $\mathbb{D}^{*(k)} = (\boldsymbol{b}^{*(k)}_1, \ldots, \boldsymbol{b}^{*(k)}_{n_k}, \boldsymbol{d}^{*(k)}_{n_k+1}, \ldots, \boldsymbol{d}^{*(k)}_{2n_k}, \boldsymbol{b}^{*(k)}_{2n_k+1}, \ldots, \boldsymbol{b}^{*(k)}_{3n_k+1})$, $\widehat{\mathbb{D}}^{(k)} = (\boldsymbol{b}^{(k)}_1, \ldots, \boldsymbol{b}^{(k)}_{n_k}, \boldsymbol{b}^{(k)}_{2n_k+1}, \ldots, \boldsymbol{b}^{(k)}_{3n_k+1})$, $k = 1, 2$.
Finally, $\mathcal{C}$ hands $(\mathsf{param}_{\overrightarrow{n}}, \widehat{\mathbb{D}}^{(0)}, \mathbb{D}^{*(0)}, \boldsymbol{y}^{*(0)}_{\beta}, \boldsymbol{f}^{(0)}, \{\widehat{\mathbb{D}}^{(k)}, \mathbb{D}^{*(k)}, \{\boldsymbol{y}^{*(k)}_{\beta,i}, \boldsymbol{f}^{(k)}_{i}\}_{i=1,\ldots,n_k}\}_{k=1,2})$ over to $\mathcal{B}$ and outputs $\beta' \in \{0, 1\}$ if $\mathcal{B}$ outputs β'.
For π in Basic Problem 2, let $\pi' = z\pi$. Then, with respect to $\pi', \mathbb{D}^{(k)}, \mathbb{D}^{*(k)}$, $k = 0, 1, 2$, the above answer to $\mathcal{B}$ has the same distribution as in the instance of Problem 2. □