Forward-Secure Hierarchical Predicate Encryption

Juan Manuel González Nieto[1], 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] Department of Computer Science, University of Surrey, United Kingdom
mark@manulis.eu

Abstract. Secrecy of decryption keys is an important pre-requisite for security of any encryption scheme. *Forward Security (FS)* reduces damage from compromised keys by guaranteeing confidentiality of messages that were encrypted prior to the compromise event. In this paper we introduce FS to the powerful setting of *Hierarchical Predicate Encryption (HPE)*, proposed by Okamoto and Takashima (Asiacrypt 2009). 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.

Keywords: Forward Security, Predicate Encryption, Inner Product.

1 Introduction

Predicate Encryption. We focus on the notion of Predicate Encryption (PE), formalized by Katz, Sahai, and Waters [21], building on Hidden Vector Encryption (HVE) [6], and further studied in [22, 24, 25, 27, 28, 33, 34]. 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) [2, 11, 15, 29] 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) [21] — are particularly powerful since they can be used to evaluate a large class of predicates, including conjunctions or disjunctions of equality tests, comparisons, and

M. Abdalla and T. Lange (Eds.): Pairing 2012, LNCS 7708, pp. 83–101, 2013.

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 product $\overrightarrow{x} \cdot \overrightarrow{y} = 0$. While the original scheme from [21] was proven to be selectively secure under non-standard assumptions, recent result of Lewko *et al.* [22] provided more sophisticated PE constructions achieving (stronger) adaptive security under non-standard assumptions. Furthermore, Okamoto and Takashima [25] investigated Functional Encryption that is adaptive security under standard assumptions. In [22, 24] 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) [5, 32] and the majority of these schemes are pairing-based.

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 [12, 16], forward security has further been explored in digital signatures [1,18] and in public key encryption (PKE) [8]; see [18] 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 [8], was built from the technical tool, called *binary tree encryption* [20], which in turn is implied by Hierarchical IBE (HIBE) [14, 17] 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.* [36] 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 [3] 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 [23]. As mentioned by Boyen and Waters [7] and also explored in [10, 13, 30, 31, 34] 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 [22,24], 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. We note that obtaining forward security in HPE schemes by applying techniques from existing FS-PKE [8] and FS-HIBE [36] results in a number of obstacles. For example, a "cross-product" combination of two HPE schemes [22, 24], akin to the case of two HIBE schemes for FS-HIBE in [36], 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.

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 [21]. Our scheme is secure (adaptively attribute-hiding) in the standard model under the well-known Decision Linear (DLIN) assumption [4] 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 [23, 36], 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 [10] 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 [35] and uses the concept of dual pairing vector spaces (DPVS) of Okamoto and Takashima [24]. Techniques underlying forward security of the scheme can be seen as reminiscent of binary tree encryption [8] 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 [36], seems not feasible with existing HPE constructions [22,24]. On a high level, we modify the existing HPE scheme from [22] 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 [8]. The modification of the scheme in [22] 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 [25], 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.

2 Background on Dual Pairing Vector Spaces and Complexity Assumption

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) [24]). *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:

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

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) [4]). *The DLIN problem is to decide on bit $\beta \in \{0, 1\}$, given the output $(\mathsf{param}_{\mathbb{G}}, G, aG, bG, acG, bdG, Y_\beta)$ of the probabilistic algorithm*

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

The advantage $\mathsf{Adv}_{\mathcal{D}}^{\mathrm{DLIN}}(\lambda)$ of a probabilistic polynomial-time DLIN solver $\mathcal{D}$ is defined as follows:

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

The DLIN assumption states that for any $\mathcal{D}$ 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 [22] to describe our HPE with inner-product predicates. 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 use 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 (FS-HPE). *A forward secure hierarchical predicate encryption scheme 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.*

$\mathsf{Delegate}(SK_{i,l}, i, \overrightarrow{x}_{l+1})$ *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*.*

$\mathsf{Update}(SK_{i,l}, i)$ *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* $SK_{i,l}$*.*

$\mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h), i, M)$ *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*.*

$\mathsf{Decrypt}(C, SK_{i,l})$ *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* $C \xleftarrow{R} \mathsf{Encrypt}(PK, (\overrightarrow{y}_1, \ldots, \overrightarrow{y}_h), i, M)$ *and* $M' = \mathsf{Decrypt}(C, SK_{i,l})$*. 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: 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* $C = \mathsf{Encrypt}(PK, Y^{(0)}, I, M^{(0)})$ *and if* $b = 1$ *then* $\mathcal{A}$ *is given* $C = \mathsf{Encrypt}(PK, Y^{(1)}, I, M^{(1)})$*.*

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* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{FS\text{-}HPE}}(\lambda) = |\mathsf{Pr}[b = b'] - 1/2|$.

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 [28] proposed a PE (HPE) which allow such key query, provided that $M^{(0)} = M^{(1)}$. The technique of Okamoto and Takashima [28] 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 [22], 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 [22] 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 [9] that uses one-time signatures to authenticate the ciphertext.

4 Our Forward-Secure HPE Scheme

High-Level Description. For simplicity of presentation, our FS-HPE makes use of a version of FS-PKE scheme by Katz [19]. In Katz's scheme, time periods are associated with the leaf nodes of a binary tree while in Canetti *et al.* scheme [8], 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 [22], 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 [22], 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 [22] 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 [22], 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.

Detailed Description. The five algorithms of our FS-HPE scheme are detailed in the following:
$\mathsf{RootSetup}\big(1^\lambda, N = 2^\kappa, \overrightarrow{\mu} = (n; d, \mu_1, \ldots, \mu_d)\big)$:
Let $\overrightarrow{x}_1$ be the root predicate and let $L = 2\kappa$ and $\overrightarrow{n} = (2; n, L)$. Compute

$$(\mathsf{param}_{\overrightarrow{n}}, \mathbb{B}^{(0)}, \mathbb{B}^{*(0)}, \mathbb{B}^{(1)}, \mathbb{B}^{*(1)}, \mathbb{B}^{(2)}, \mathbb{B}^{*(2)}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}),$$
$$\widetilde{\mathbb{B}}^{(0)} = (\boldsymbol{b}_1^{(0)}, \boldsymbol{b}_3^{(0)}, \boldsymbol{b}_5^{(0)}),\ \widetilde{\mathbb{B}}^{(1)} = (\boldsymbol{b}_1^{(1)}, \ldots, \boldsymbol{b}_n^{(1)}, \boldsymbol{b}_{3n+1}^{(1)}),\ \widetilde{\mathbb{B}}^{(2)} = (\boldsymbol{b}_1^{(2)}, \ldots, \boldsymbol{b}_L^{(2)}, \boldsymbol{b}_{3L+1}^{(2)}),$$
$$\widetilde{\mathbb{B}}^{*(0)} = (\boldsymbol{b}_1^{*(0)}, \boldsymbol{b}_3^{*(0)}),\ \widetilde{\mathbb{B}}^{*(1)} = (\boldsymbol{b}_1^{*(1)}, \ldots, \boldsymbol{b}_n^{*(1)}),\ \widetilde{\mathbb{B}}^{*(2)} = (\boldsymbol{b}_1^{*(2)}, \ldots, \boldsymbol{b}_L^{*(2)}),$$
$$\widehat{\mathbb{B}}^{*(1)} = (\boldsymbol{b}_{2n+1}^{*(1)}, \ldots, \boldsymbol{b}_{3n}^{*(1)}),\ \widehat{\mathbb{B}}^{*(2)} = (\boldsymbol{b}_{2L+1}^{*(2)}, \ldots, \boldsymbol{b}_{3L}^{*(2)}).$$

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 $\psi, \psi', \alpha_{\mathsf{dec}}, \alpha_{\mathsf{dec}}^{(1)}, \alpha_{\mathsf{dec}}^{(2)} \xleftarrow{\mathsf{U}} \mathbb{F}_q$ such that $\alpha_{\mathsf{dec}} = \alpha_{\mathsf{dec}}^{(1)} + \alpha_{\mathsf{dec}}^{(2)}$. Pick $\eta_{\mathsf{dec}}^{(0)}, \beta_{\mathsf{dec},1}^{(1)}, \beta_{\mathsf{dec},1}^{(2)}, \beta_{\mathsf{ran},j,1}^{(1)} (j = 1, 2), \beta_{\mathsf{ran},j,1}^{(2)} (j = 1, 2), \beta_{\mathsf{del},j,1}^{(1)} (j = 1, \ldots, n)$, $\beta_{\mathsf{del},j,1}^{(2)} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q$, $\overrightarrow{\eta}_{\mathsf{dec}}^{(2)}, \overrightarrow{\eta}_{\mathsf{ran},j}^{(2)} (j = 1, 2), \overrightarrow{\eta}_{\mathsf{del},j}^{(2)} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathbb{F}_q^L$,

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

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

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

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

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

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

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

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

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

$\mathsf{ComputeNext}(PK, sk_{w,l}, w)$: 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 $(\overrightarrow{x}_1, \ldots, \overrightarrow{x}_l)$. 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 $t = 1, \ldots, r+1$. $\boldsymbol{r}^{(1)}_{\mathsf{dec}}, \boldsymbol{r}^{(1)}_{\mathsf{ran},j} (j = 1, \ldots, l+1), \boldsymbol{r}^{(1)}_{\mathsf{del},j} (j = 1, \ldots, n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1}, \ldots, \boldsymbol{b}^{*(1)}_{3n} \rangle$, $\boldsymbol{r}^{(2)}_{\mathsf{dec}}, \boldsymbol{r}^{(2)}_{\mathsf{ran},j} (j = 1, \ldots, r+2), \boldsymbol{r}^{(2)}_{\mathsf{del},j} (j = 1, \ldots, L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1}, \ldots, \boldsymbol{b}^{*(2)}_{3L} \rangle$.
Compute

$$\boldsymbol{k}^{(0)}_{w0,l,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \epsilon^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_4,$$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{dec}} = \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(1)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(2)}_{w0,l,\mathsf{dec}} = \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{dec}} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \boldsymbol{r}^{(2)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{ran},j} = \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{r}^{(1)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, l+1,$$

$$\boldsymbol{k}^{(2)}_{w0,l,\mathsf{ran},j} = \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{ran},j} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \boldsymbol{r}^{(2)}_{\mathsf{ran},j}, \text{ for } j = 1, \ldots, r+2,$$

$$\boldsymbol{k}^{(1)}_{w0,l,\mathsf{del},j} = \sum_{t=1}^{l+1} \epsilon^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \psi \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(1)}_{\mathsf{del},j}, \text{ for } j = \mu_l + 1, \ldots, n,$$

$$\boldsymbol{k}^{(2)}_{w0,l,\mathsf{del},j} = \sum_{t=1}^{r+1} \epsilon^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \sigma_{\mathsf{del},j} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},2(r+1)} + \psi' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{r}^{(2)}_{\mathsf{del},j},$$
$$\text{for } j = 2(r+1)+1, \ldots, L.$$

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

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

$$\boldsymbol{k}^{(0)}_{w1,l,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \varepsilon^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_{4},$$

$$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{dec}} = \boldsymbol{k}^{(1)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{l+1} \varepsilon^{(1)}_{\mathsf{dec},t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{t}^{(1)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(2)}_{w1,l,\mathsf{dec}} = \boldsymbol{k}^{(2)}_{w,l,\mathsf{dec}} + \sum_{t=1}^{r+1} \varepsilon^{(2)}_{\mathsf{dec},t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{dec}} \left(\sum_{i=2r+1}^{2r+2} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i} \right) + \boldsymbol{t}^{(2)}_{\mathsf{dec}},$$

$$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{ran},j} = \sum_{t=1}^{l+1} \varepsilon^{(1)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \boldsymbol{t}^{(1)}_{\mathsf{ran},j}, \text{ for } j=1,\ldots,l+1,$$

$$\boldsymbol{k}^{(2)}_{w1,l,\mathsf{ran},j} = \sum_{t=1}^{r+1} \varepsilon^{(2)}_{\mathsf{ran},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{ran},j} \left(\sum_{i=2r+1}^{2r+2} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i} \right) + \boldsymbol{t}^{(2)}_{\mathsf{ran},j}, \text{ for } j=1,\ldots,r+2,$$

$$\boldsymbol{k}^{(1)}_{w1,l,\mathsf{del},j} = \sum_{t=1}^{l+1} \varepsilon^{(1)}_{\mathsf{del},j,t} \boldsymbol{k}^{(1)}_{w,l,\mathsf{ran},t} + \tau \boldsymbol{k}^{(1)}_{w,l,\mathsf{del},j} + \boldsymbol{t}^{(1)}_{\mathsf{del},j}, \text{ for } j=\mu_l+1,\ldots,n,$$

$$\boldsymbol{k}^{(2)}_{w1,l,\mathsf{del},j} = \sum_{t=1}^{r+1} \varepsilon^{(2)}_{\mathsf{del},j,t} \boldsymbol{k}^{(2)}_{w,l,\mathsf{ran},t} + \varsigma_{\mathsf{del},j} \left(\sum_{i=2r+1}^{2r+2} \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},i} \right) + \tau' \boldsymbol{k}^{(2)}_{w,l,\mathsf{del},j} + \boldsymbol{t}^{(2)}_{\mathsf{del},j}, \text{ for } j=2(r+1)+1,\ldots,L.$$

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

$\mathsf{Delegate}(SK_{i,l}, i, \overrightarrow{x}_{l+1} = (x_{\mu_l+1},\ldots,x_{\mu_{l+1}}))$: Parse i as $i_1,\ldots,i_\kappa$ where $\kappa = \log_2 N$. Parse $SK_{i,l}$ as $(sk_{i,l}, \{sk_{i|_{k-1},l}\}_{i_k=0})$. 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 $t=1,\ldots,r+1$. $\boldsymbol{r}^{(1)}_{\mathsf{dec}}, \boldsymbol{r}^{(1)}_{\mathsf{ran},j} (j=1,\ldots,l+2), \boldsymbol{r}^{(1)}_{\mathsf{del},j} (j=1,\ldots,n) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(1)}_{2n+1},\ldots,\boldsymbol{b}^{*(1)}_{3n}\rangle$, $\boldsymbol{r}^{(2)}_{\mathsf{dec}}, \boldsymbol{r}^{(2)}_{\mathsf{ran},j} (j=1,\ldots,r+1), \boldsymbol{r}^{(2)}_{\mathsf{del},j} (j=1,\ldots,L) \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}^{*(2)}_{2L+1},\ldots,\boldsymbol{b}^{*(2)}_{3L}\rangle$. Compute

$$\boldsymbol{k}^{(0)}_{w,l+1,\mathsf{dec}} = \boldsymbol{k}^{(0)}_{w,l,\mathsf{dec}} + \gamma^{(0)}_{\mathsf{dec}} \boldsymbol{b}^{*(0)}_{4},$$

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

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

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

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

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

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

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.

$\mathsf{Update}(SK_{i,l}, i)$: This algorithm follows the concept from [8, 19] 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 $sk_{(i'0^{d-\tilde{k}},l)}$. (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.) Erase $sk_{i',l}$ and output the remaining keys as $SK_{(i+1),l}$.

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

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

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

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

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

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

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 $1 \le i \le l$, and C and $SK_{i,l}$ are encoded with the same time period i then M can be recovered by computing $\boldsymbol{c}^{(M)} / e(\boldsymbol{c}^{(0)}, \boldsymbol{k}^{(0)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(1)}, \boldsymbol{k}^{(1)}_{i,l,\mathsf{dec}}) e(\boldsymbol{c}^{(2)}, \boldsymbol{k}^{(2)}_{i,l,\mathsf{dec}})$, since

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

Remark 4. Recently, Okamoto and Takashima [27] proposed a PE with short secret keys. We note that their scheme can be easily applied to our system to achieve better efficiency in key size. Moreover, in an updated version [26], Okamoto and Takashima devised a payload-hiding HIPE with compact secret keys. The technique [26] can also be applied in our system, specifically, for the time period subtree.

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 PPT machine $\mathcal{D}$ such that for any security parameter λ,*

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

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 provided in the full version.

5 Conclusion

In this paper, we introduced the notion of forward security to the 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 at which 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. Using level-1 hierarchy we obtain first adaptively-secure FS-PE/ABE construction. By setting the inner-product predicate to perform the equality test, we achieve the first adaptively-secure anonymous FS-HIBE scheme under the DLIN assumption.

References

1. Bellare, M., Miner, S.K.: A Forward-Secure Digital Signature Scheme. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 431–448. Springer, Heidelberg (1999)
2. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy, pp. 321–334. IEEE Computer Society (2007)
3. Boneh, D., Boyen, X., Goh, E.-J.: Hierarchical Identity Based Encryption with Constant Size Ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
4. Boneh, D., Boyen, X., Shacham, H.: Short Group Signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
5. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
6. Boneh, D., Waters, B.: Conjunctive, Subset, and Range Queries on Encrypted Data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
7. Boyen, X., Waters, B.: Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
8. Canetti, R., Halevi, S., Katz, J.: A Forward-Secure Public-Key Encryption Scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
9. Canetti, R., Halevi, S., Katz, J.: Chosen-Ciphertext Security from Identity-Based Encryption. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 207–222. Springer, Heidelberg (2004)
10. De Caro, A., Iovino, V., Persiano, G.: Fully Secure Anonymous HIBE and Secret-Key Anonymous IBE with Short Ciphertexts. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 347–366. Springer, Heidelberg (2010)
11. Chase, M.: Multi-authority Attribute Based Encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
12. Diffie, W., Van Oorschot, P.C., Wiener, M.J.: Authentication and authenticated key exchanges. Designs, Codes and Cryptography 2, 107–125 (1992)
13. Ducas, L.: Anonymity from Asymmetry: New Constructions for Anonymous HIBE. In: Pieprzyk, J. (ed.) CT-RSA 2010. LNCS, vol. 5985, pp. 148–164. Springer, Heidelberg (2010)
14. Gentry, C., Silverberg, A.: Hierarchical ID-Based Cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)

15. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: ACM CCS 2006, pp. 89–98. ACM (2006)
16. Günther, C.G.: An Identity-Based Key-Exchange Protocol. In: Quisquater, J.-J., Vandewalle, J. (eds.) EUROCRYPT 1989. LNCS, vol. 434, pp. 29–37. Springer, Heidelberg (1990)
17. Horwitz, J., Lynn, B.: Toward Hierarchical Identity-Based Encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)
18. Itkis, G., Reyzin, L.: Forward-Secure Signatures with Optimal Signing and Verifying. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 332–354. Springer, Heidelberg (2001)
19. Katz, J.: A forward-secure public-key encryption scheme. Cryptology ePrint Archive, Report 2002/060 (2002), http://eprint.iacr.org/
20. Katz, J.: Binary Tree Encryption: Constructions and Applications. In: Lim, J.-I., Lee, D.-H. (eds.) ICISC 2003. LNCS, vol. 2971, pp. 1–11. Springer, Heidelberg (2004)
21. Katz, J., Sahai, A., Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
22. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
23. Lewko, A., Waters, B.: New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 455–479. Springer, Heidelberg (2010)
24. Okamoto, T., Takashima, K.: Hierarchical Predicate Encryption for Inner-Products. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 214–231. Springer, Heidelberg (2009)
25. Okamoto, T., Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
26. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. Cryptology ePrint Archive, Report 2010/563 (2010), http://eprint.iacr.org/
27. Okamoto, T., Takashima, K.: Achieving Short Ciphertexts or Short Secret-Keys for Adaptively Secure General Inner-Product Encryption. In: Lin, D., Tsudik, G., Wang, X. (eds.) CANS 2011. LNCS, vol. 7092, pp. 138–159. Springer, Heidelberg (2011)
28. Okamoto, T., Takashima, K.: Adaptively Attribute-Hiding (Hierarchical) Inner Product Encryption. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 591–608. Springer, Heidelberg (2012)
29. Sahai, A., Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
30. Seo, J.H., Cheon, J.H.: Fully secure anonymous hierarchical identity-based encryption with constant size ciphertexts. Cryptology ePrint Archive, Report 2011/021 (2011), http://eprint.iacr.org/
31. Seo, J.H., Kobayashi, T., Ohkubo, M., Suzuki, K.: Anonymous Hierarchical Identity-Based Encryption with Constant Size Ciphertexts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 215–234. Springer, Heidelberg (2009)

32. Shamir, A.: Identity-Based Cryptosystems and Signature Schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
33. Shen, E., Shi, E., Waters, B.: Predicate Privacy in Encryption Systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)
34. Shi, E., Waters, B.: Delegating Capabilities in Predicate Encryption Systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
35. Waters, B.: Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
36. Yao, D., Fazio, N., Dodis, Y., Lysyanskaya, A.: Id-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In: ACM CCS 2004, pp. 354–363. ACM (2004)