Identity-based Hierarchical Key-insulated Encryption without Random Oracles*

Yohei Watanabe[1,3] and Junji Shikata[1,2]

[1] Graduate School of Environment and Information Sciences, Yokohama National University, Japan
[2] Institute of Advanced Sciences, Yokohama National University, Japan
[3] Information Technology Research Institute, AIST, Japan
watanabe-yohei-xs@ynu.jp, shikata@ynu.ac.jp

Abstract. Key-insulated encryption is one of the effective solutions to a key exposure problem. Recently, identity-based encryption (IBE) has been used as one of fundamental cryptographic primitives in a wide range of various applications, and it is considered that the identity-based key-insulated security has a huge influence on the resulting applications. At Asiacrypt'05, Hanaoka et al. proposed an identity-based hierarchical key-insulated encryption (hierarchical IKE) scheme. Although their scheme is secure in the random oracle model, it has a "hierarchical key-updating structure," which is attractive functionality that enhances key exposure resistance.
In this paper, we first propose the hierarchical IKE scheme without random oracles. Our hierarchical IKE scheme is secure under the symmetric external Diffie–Hellman (SXDH) assumption, which is known as the simple and static one. Furthermore, when the hierarchy depth is one (i.e. not hierarchical case), our scheme is the first IKE scheme that achieves constant-size parameters including public parameters, secret keys, and ciphertexts.

Keywords: Key-insulated encryption, identity-based hierarchical key-insulated encryption, hierarchical identity-based encryption, asymmetric pairing.

1 Introduction

1.1 Background

A key exposure problem is unavoidable since human errors cannot seem to be eliminated in the future, and many researchers have tackled this problem in modern cryptography area so far. *Key-insulation*, which is introduced by Dodis et al. [12], is one solution to this problem. Specifically, they proposed public key encryption with the key-insulated property, which is called *public-key-based*

key-insulated encryption (PK-KIE). In PK-KIE, a user has two kinds of secret keys, so-called *a decryption key* and *a helper key*. The decryption key is used for decrypting ciphertexts and assumed to be stored in a powerful but insecure device such as laptops and smartphones. Meanwhile, the helper key is used for updating the decryption key and assumed to be stored in a physically-secure but computationally-limited device such as USB pen drives. Traditionally, in key-insulated cryptography, the following two kinds of security notions are considered:

1. If a number of decryption keys are exposed, the fact does not affect decryption keys at other time-periods.
2. Even if a helper key is exposed, the security is not compromised unless at least one decryption key is exposed.

We say a key-insulated system is secure if it satisfies 1; and it is *strongly* secure if it satisfies both 1 and 2. Specifically, the lifetime of the system is divided into discrete time-periods, and the user can decrypt the ciphertext encrypted at some time-period t by using a decryption key updated at the same time-period t. Therefore, even if the decryption key at t is exposed, the fact does not affect decryption keys at other time-periods, and hence the impact of the exposure can be significantly reduced.

Following a seminal work by Dodis et al. [12], symmetric-key-based key-insulated encryption [14], key-insulated signatures [13], and parallel key-insulated encryption [17, 18, 23] have been proposed so far. In addition to key-insulated cryptography, researchers have tackled the key exposure problem in various flavors. In forward-secure cryptography [1, 8], users update their own secret keys at the beginning of each time-period. Even if the secret key is exposed, an adversary cannot get any information of ciphertexts encrypted at previous time-periods. Intrusion-resilient cryptography [10, 11, 20] realizes both key-insulated security and forward security simultaneously at the sacrifice of efficiency and practicality.

In this paper, we focus on the key-insulation paradigm in the identity-based setting. Since identity-based encryption (IBE) has been used as one of fundamental cryptographic primitives in a wide range of various applications, we believe that the identity-based key-insulated security has a huge influence on the resulting applications. Also, developing key-insulated cryptography in the identity-based area is the first step to consider the key-insulated security in the attribute-based [3, 26] and functional encryption [7] settings. Thus, we consider that it is important to consider the identity-based key-insulated security. However, in the IBE context, there are only few researches on key-insulation. Hanaoka et al. [19] proposed the first identity-based (hierarchical) key-insulated encryption (IKE) scheme in the random oracle model. In their hierarchical IKE scheme, the key-updating mechanism has the hierarchical structure (and the scheme does not have a delegating property). Namely, not only a decryption key but also a helper key can be updated by a higher-level helper key. Since this "hierarchy" is not the same as that of hierarchical IBE (HIBE) [16], only applying techniques used in the HIBE context is insufficient for constructing secure (in particular, *strongly* secure) IKE schemes. The hierarchical property is attractive

since it enhances resistance to key exposure and there seem to be various applications due to progress in information technology (e.g., the popularization of smartphones). Let us consider an example: Suppose that each employee has a smartphone for business use, a laptop, and a PC at his office. A decryption key is stored in the smartphone, and it is updated by a 1-st level helper key stored in his laptop every day. However, the 1-st level helper key might be leaked since he carries around the laptop, and connects to the Internet via the laptop. Thus, the 1-st level helper key is also updated by a 2-nd level helper key stored in his PC every two–three months. Since the PC is not completely isolated from the Internet, every half a year, his boss updates the 2-nd level helper key is updated by 3-rd level helper key stored in an isolated private device. Thus, we believe hierarchical IKE has many potential applications.

After the proposal of hierarchical IBE by Hanaoka et al., two (not hierarchical) IKE schemes with additional properties in the standard model were proposed. One is the so-called *parallel* IKE scheme, which was proposed by Weng et al. [31]. The other is the so-called *threshold* IKE scheme, which was proposed by Weng et al. [32]. These two schemes enhance the resistance to helper key exposure by splitting a helper key into multiple ones. However, once the (divided) helper key is leaked, the security cannot be recovered. We now emphasize that the hierarchical key-updating structure is useful since even if some helper key is exposed, the helper key can be updated. However, there have been no hierarchical IKE schemes without random oracles so far.

1.2 Our Contribution

In this paper, our aim is to construct a hierarchical IKE scheme such that: (1) we can prove the security in the standard model from simple computational assumptions; and (2) when the hierarchy depth is one (i.e., not hierarchical case), the scheme achieves all constant-size parameters including public parameters, secret keys, and ciphertexts.

As a result, we propose the first hierarchical IKE scheme in the standard model. Specifically, we construct the hierarchical IKE scheme from the symmetric external Diffie–Hellman (SXDH) assumption, which is a static and simple one. Further, the proposed scheme achieves the constant-size parameters when the hierarchy is one, whereas public parameters of the (not hierarchical) existing scheme [32] depend on sizes of identity spaces (also see Section 4.1 for comparison). This is due to differences of base IBE schemes of each scheme. Our (hierarchical) IKE scheme is based on the Jutla–Roy IBE [22] and its variant [25], whereas the existing scheme (but not hierarchical one) [32] is based on the Waters IBE [29]. In the following, we explain why a naive solution is insufficient and why achieving (1) and (2) is challenging.

Why a (trivial) hierarchical IKE scheme from HIBE is insufficient.[4] One may think that a hierarchical IKE scheme can be easily obtained from

[4] This fact was also mentioned in [19].

an arbitrary HIBE scheme. However, the resulting IKE scheme is insecure in our security model, which was first formalized in [19]. The reason for this is that our security model includes the *strong* security model, and hence the fact makes a hierarchical IKE scheme from HIBE insecure. More specifically, a trivial construction is as follows. Let $sk_{\mathtt{I}}$ be a secret key for some identity $\mathtt{I}$ in HIBE, and $hk_{\mathtt{I}}^{(\ell)}$ be an ℓ-th level helper key for $\mathtt{I}$ in IKE. We set $sk_{\mathtt{I}}$ as $hk_{\mathtt{I}}^{(\ell)}$, and lower-level helper and decryption keys can be obtained from $sk_{\mathtt{I}}$ by regarding time-periods as descendants' identity. However, it is easy to see that if ℓ-th level helper key is exposed, then an adversary can obtain all lower-level keys, and thus, the resulting scheme does not meet the strong security. In fact, Bellare and Palacio [2] showed that *not strongly secure* PK-KIE is equivalent to IBE for a similar reason.

Difficulties in constructing a constant-size IKE scheme from simple computational assumptions. The main difficulty in constructing an IKE scheme is that an adversary can get various keys regarding a target identity $\mathtt{I}^*$, whereas in (H)IBE, the adversary cannot get any information on a secret key for $\mathtt{I}^*$. This point makes a construction methodology non-trivial. Actually, it seems difficult to apply the Waters dual-system IBE [30] (and its variant [24]) as the underlying basis of IKE schemes. Technically, in their scheme each of secret keys and ciphertexts contains some random exponent, so-called tag_K and tag_C, respectively. In their proof, these tags for some $\mathtt{I}$ are needed to be generated by inputting $\mathtt{I}$ into some pairwise independent function, which is embedded into public parameters in advance. This generating procedure is necessary for cancellation of values and hence the security proof. Although it holds $tag_K = tag_C$ for the same identity $\mathtt{I}$, the proof works well since it is enough to generate only tag_K for all identities $\mathtt{I} \neq \mathtt{I}^*$ and only tag_C for the target identity $\mathtt{I}^*$. However, in the IKE setting, not only tag_C but also tag_K for $\mathtt{I}^*$ have to be generated since an adversary can get leaked decryption and helper keys for $\mathtt{I}^*$, and hence, the proof does not go well. To overcome this challenging point, we set (the variant of) the Jutla–Roy IBE [22, 25], which is another type of constant-size IBE schemes, as the basis of our IKE scheme, and thus we can realize the first constant-size IKE scheme under the SXDH assumption. Further, we can also obtain the hierarchical IKE scheme by extending the technique into the hierarchical setting.

Organization of this paper. In Section 2, we describe the notation used in this paper, asymmetric pairings, complexity assumptions, and functions which map time to discrete time-periods. In Section 3, we give a model and security definition of hierarchical IKE. In Section 4, we propose a direct construction of our hierarchical IKE scheme, and give the efficiency comparison among our scheme and existing schemes. In Section 5, we show the security proof of our scheme. In Section 6, we show a CCA-secure hierarchical IKE scheme. In Section 7, we conclude this paper.

2 Preliminaries

Notation. In this paper, "probabilistic polynomial-time" is abbreviated as "PPT". Let $\mathbb{Z}_p := \{0, 1, \ldots, p-1\}$ and $\mathbb{Z}_p^\times := \mathbb{Z}_p \setminus \{0\}$. If we write $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{A}(x_1, x_2, \ldots, x_n)$ for an algorithm $\mathcal{A}$ having n inputs and m outputs, it means to input $x_1, x_2, \ldots, x_n$ into $\mathcal{A}$ and to get the resulting output $y_1, y_2, \ldots, y_m$. We write $(y_1, y_2, \ldots, y_m) \leftarrow \mathcal{A}^{\mathcal{O}}(x_1, x_2, \ldots, x_n)$ to indicate that an algorithm $\mathcal{A}$ that is allowed to access an oracle $\mathcal{O}$ takes $x_1, x_2, \ldots, x_n$ as input and outputs $(y_1, y_2, \ldots, y_m)$. If $\mathcal{X}$ is a set, we write $x \xleftarrow{\$} \mathcal{X}$ to mean the operation of picking an element x of $\mathcal{X}$ uniformly at random. We use λ as a security parameter. $\mathcal{M}$ and $\mathcal{I}$ denote sets of plaintexts and IDs, respectively, which are determined by a security parameter λ.

Bilinear Group. A bilinear group generator $\mathcal{G}$ is an algorithm that takes a security parameter λ as input and outputs a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where p is a prime, $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are multiplicative cyclic groups of order p, g_1 and g_2 are (random) generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively, and e is an efficiently computable and non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ with the following bilinear property: For any $u, u' \in \mathbb{G}_1$ and $v, v' \in \mathbb{G}_2$, $e(uu', v) = e(u, v)e(u', v)$ and $e(u, vv') = e(u, v)e(u, v')$, and for any $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$ and any $a \in \mathbb{Z}_p$, $e(u^a, v) = e(u, v^a) = e(u, v)^a$.

A bilinear map e is called symmetric or a "Type-1" pairing if $\mathbb{G}_1 = \mathbb{G}_2$. Otherwise, it is called asymmetric. In the asymmetric setting, e is called a "Type-2" pairing if there is an efficiently computable isomorphism either from $\mathbb{G}_1$ to $\mathbb{G}_2$ or from $\mathbb{G}_2$ to $\mathbb{G}_1$. If no efficiently computable isomorphisms are known, then it is called a "Type-3" pairing. In this paper, we focus on the Type-3 pairing, which is the most efficient setting (For details, see [9, 15]).

Symmetric External Diffie–Hellman (SXDH) Assumption. We give the definition of the decisional Diffie–Hellman (DDH) assumption in $\mathbb{G}_1$ and $\mathbb{G}_2$, which are called the DDH1 and DDH2 assumptions, respectively.

Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDH1 problem as follows.

$$Adv_{\mathcal{G},\mathcal{A}}^{DDH1}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}, \\ c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p,\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } T := g_1^{c_1 c_2}, \\ \text{else } T \xleftarrow{\$} \mathbb{G}_1, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1, g_2, g_1^{c_1}, g_1^{c_2}, T) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 1 (DDH1 Assumption). *The DDH1 assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv_{\mathcal{G},\mathcal{A}}^{DDH1}(\lambda)$ is negligible in λ.*

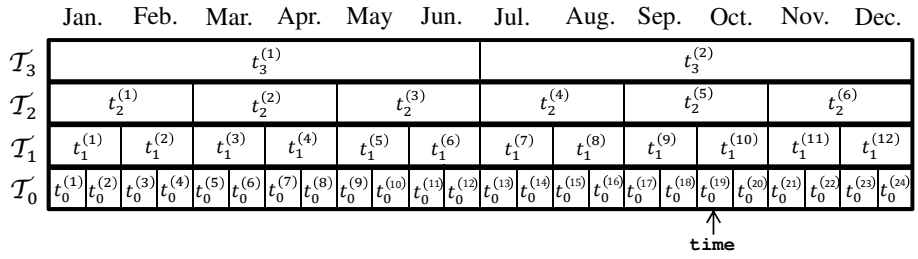

Fig. 1. Intuition of time-period map functions.

Similarly, we define the DDH2 problem. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DDH2 problem as follows.

$$Adv_{\mathcal{G},\mathcal{A}}^{DDH2}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} D := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow \mathcal{G}, \\ c_1, c_2 \xleftarrow{\$} \mathbb{Z}_p,\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 0 \text{ then } T := g_2^{c_1 c_2}, \\ \text{else } T \xleftarrow{\$} \mathbb{G}_2, \\ b' \leftarrow \mathcal{A}(\lambda, D, g_1, g_2, g_2^{c_1}, g_2^{c_2}, T) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 2 (DDH2 Assumption). *The DDH2 assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv_{\mathcal{G},\mathcal{A}}^{DDH2}(\lambda)$ is negligible in λ.*

Definition 3 (SXDH Assumption). *We say that the SXDH assumption relative to a generator $\mathcal{G}$ holds if both the DDH1 and DDH2 assumptions relative to $\mathcal{G}$ hold.*

Time-period Map Functions. In this paper, we deal with *several kinds of time-periods* since we consider that update intervals of each level key are different. For example, in some practical applications, it might be suitable that a decryption key (i.e. 0-th level key) and a 1-st level helper key should be updated every day and every three months, respectively. To describe such different update intervals of each level key, we use functions, which is so-called *time-period map functions*. This functions were also used in [19]. Now, let $\mathcal{T}$ be a (possibly infinite) set of *time*, and $\mathcal{T}_j$ $(0 \leq j \leq \ell - 1)$ be a finite set of *time-periods*. We assume $|\mathcal{T}_0| \geq |\mathcal{T}_1| \geq \cdots \geq |\mathcal{T}_{\ell-1}|$. This means that a lower-level key is updated more frequently than the higher-level keys. Then, we assume there exists a function T_j $(0 \leq j \leq \ell - 1)$ which map time $\mathtt{time} \in \mathcal{T}$ to a time-period $t_j \in \mathcal{T}_j$. For the understanding of readers, by letting $\mathtt{time} = \mathtt{9:59/7th/Oct./2015}$ and $\ell := 4$, we give an example in Figure 1 and below. For example, we have $T_0(\mathtt{time}) = t_0^{(19)} = \mathtt{1st\text{-}15th/Oct./2015}$, $T_1(\mathtt{time}) = t_1^{(10)} = \mathtt{Oct./2015}$, $T_2(\mathtt{time}) = t_2^{(5)} = \mathtt{Oct.\text{-}Dec./2015}$, and $T_3(\mathtt{time}) = t_3^{(2)} = \mathtt{Jul.\text{-}Dec./2015}$. Namely, in this example, it is assumed that the decryption key, and 1-st, 2-nd, and 3-rd helper keys are updated every half a month, every month, every three months, and every half a year. Further, we can also define a function T_ℓ such that $T_\ell(\mathtt{time}) = 0$ for all $\mathtt{time} \in \mathcal{T}$.

3 Identity-based Hierarchical Key-insulated Encryption

3.1 The Model

In ℓ-level hierarchical IKE, a key generation center (KGC) generates an initial decryption key $dk_{\mathtt{I},0}$ and ℓ initial helper keys $hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}$ as a secret key for a user $\mathtt{I}$. Suppose that all time-period map functions $T_0, \ldots, T_{\ell-1}$ are available to all users. The key-updating procedure when the user wants to get a decryption key at current time $\mathtt{time} \in \mathcal{T}$ from the initial helper keys is as follows. The ℓ-th level helper key $hk_{\mathtt{I},0}^{(\ell)}$ is a long-term one and is never updated. First, the user generates *key update* $\delta_{t_{\ell-1}}^{(\ell-1)}$ for the $(\ell-1)$-th level helper key from $hk_{\mathtt{I},0}^{(\ell)}$ and a time-period $t_{\ell-1} := T_{\ell-1}(\mathtt{time}) \in \mathcal{T}_{\ell-1}$. Then, the $(\ell-1)$-th level helper key $hk_{\mathtt{I},0}^{(\ell-1)}$ can be updated by the key update $\delta_{t_{\ell-1}}^{(\ell-1)}$, and the user get the helper key $hk_{\mathtt{I},t_{\ell-1}}^{(\ell-1)}$ at the time-period $t_{\ell-1}$. Similarly, the i-th level helper key $hk_{\mathtt{I},t_i}^{(i)}$ at the time-period $t_i := T_i(\mathtt{time}) \in \mathcal{T}_i$ can be obtained from $hk_{\mathtt{I},0}^{(i)}$ and $\delta_{t_i}^{(i)}$, where $\delta_{t_i}^{(i)}$ is generated from the $(i+1)$-th level helper key $hk_{\mathtt{I},t_{i+1}}^{(i+1)}$. The user can finally get the decryption key $dk_{\mathtt{I},t_0}$ at a time-period $t_0 := T_0(\mathtt{time}) \in \mathcal{T}_0$ from the 1-st level helper key $hk_{\mathtt{I},T_1(\mathtt{time})}^{(1)}$. Anyone can encrypt a plaintext M with the identity $\mathtt{I}$ and current time $\mathtt{time}^*$, and the user can decrypt the ciphertext C with his decryption key $dk_{\mathtt{I},t_0}$ only if $t_0 = T_0(\mathtt{time}^*)$. At $\mathtt{time}' \in \mathcal{T}$, the user can update the time-period of the decryption key from any time-period t_0 to $t_0' := T_0(\mathtt{time}') \in \mathcal{T}_0$ by using key update $\delta_{T_0(\mathtt{time}')}^{(0)}$. The key update $\delta_{T_0(\mathtt{time}')}^{(0)}$ can be obtained from $hk_{\mathtt{I},t_1'}^{(1)}$ only if $t_1' = T_1(\mathtt{time}')$. If not, it is necessary to get $\delta_{T_1(\mathtt{time}')}^{(1)}$ and update $hk_{\mathtt{I},t_1'}^{(1)}$. In this manner, the decryption and helper keys are updated.

An ℓ-level hierarchical IKE scheme Π_{IKE} consists of six-tuple algorithms (PGen, Gen, Δ-Gen, Upd, Enc, Dec) defined as follows. For simplicity, we omit a public parameter in the input of all algorithms except for the PGen algorithm.

- $(pp, mk) \leftarrow \mathsf{PGen}(\lambda, \ell)$: A probabilistic algorithm for parameter generation. It takes a security parameter λ and the maximum hierarchy depth ℓ as input, and outputs a public parameter pp and a master key mk.
- $(dk_{\mathtt{I},0}, hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}) \leftarrow \mathsf{Gen}(mk, \mathtt{I})$: An algorithm for user key generation. It takes mk and an identity $\mathtt{I} \in \mathcal{I}$ as input, and outputs an initial secret key $dk_{\mathtt{I},0}$ associated with $\mathtt{I}$ and initial helper keys $hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}$, where $hk_{\mathtt{I},0}^{(i)}$ $(1 \le i \le \ell)$ is assumed to be stored user's i-th level private device.
- $\delta_{T_{i-1}(\mathtt{time})}^{(i-1)}$ or $\perp \leftarrow \Delta\text{-}\mathsf{Gen}(hk_{\mathtt{I},t_i}^{(i)}, \mathtt{time})$: An algorithm for key update generation. It takes an i-th helper key $hk_{\mathtt{I},t_i}^{(i)}$ at a time period $t_i \in \mathcal{T}_i$ and current time $\mathtt{time}$ as input, and outputs key update $\delta_{T_{i-1}(\mathtt{time})}^{(i-1)}$ if $t_i = T_i(\mathtt{time})$; otherwise, it outputs $\perp$.
- $hk_{\mathtt{I},\tau_i}^{(i)} \leftarrow \mathsf{Upd}(hk_{\mathtt{I},t_i}^{(i)}, \delta_{\tau_i}^{(i)})$: A probabilistic algorithm for decryption key generation. It takes an i-th helper key $hk_{\mathtt{I},t_i}^{(i)}$ at a time-period $t_i \in \mathcal{T}_i$ and key

update $\delta_{\tau_i}^{(i)}$ at a time-period $\tau \in \mathcal{T}_i$ as input, and outputs a renewal i-th helper key $hk_{\mathtt{I},\tau_i}^{(i)}$ at τ. Note that for any $t_0 \in \mathcal{T}_0$, $hk_{\mathtt{I},t_0}^{(0)}$ means $dk_{\mathtt{I},t_0}$.
- $\langle C, \mathtt{time} \rangle \leftarrow \mathsf{Enc}(\mathtt{I}, \mathtt{time}, M)$: A probabilistic algorithm for encryption. It takes an identity $\mathtt{I}$, current time $\mathtt{time}$, and a plaintext $M \in \mathcal{M}$ as input, and outputs a pair of a ciphertext and current time $\langle C, \mathtt{time} \rangle$.
- M or $\perp \leftarrow \mathsf{Dec}(dk_{\mathtt{I},t_0}, \langle C, \mathtt{time} \rangle)$: A deterministic algorithm for decryption. It takes $dk_{\mathtt{I},t_0}$ and $\langle C, \mathtt{time} \rangle$ as input, and outputs M or $\perp$, where $\perp$ indicates decryption failure.

In the above model, we assume that Π_{IKE} meets the following correctness property: For all security parameter λ, all $\ell := poly(\lambda)$, all $(mk, pp) \leftarrow \mathsf{PGen}(\lambda, \ell)$, all $M \in \mathcal{M}$, all $(dk_{\mathtt{I},0}, hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)}) \leftarrow \mathsf{Gen}(mk, \mathtt{I})$, and all $\mathtt{time} \in \mathcal{T}$, it holds that $M \leftarrow \mathsf{Dec}(dk_{\mathtt{I},T_0(\mathtt{time})}, \mathsf{Enc}(\mathtt{I}, \mathtt{time}, M))$, where $dk_{\mathtt{I},T_0(\mathtt{time})}$ is generated as follows: For $i = \ell, \ldots, 1$, $hk_{\mathtt{I},T_{i-1}(\mathtt{time})}^{(i-1)} \leftarrow \mathsf{Upd}(hk_{\mathtt{I},t_{i-1}}^{(i-1)}, \Delta\text{-}\mathsf{Gen}(hk_{\mathtt{I},T_i(\mathtt{time})}^{(i)}, \mathtt{time}))$, where some $t_i \in \mathcal{T}_i$ and $hk_{\mathtt{I},T_0(\mathtt{time})}^{(0)} := dk_{\mathtt{I},T_0(\mathtt{time})}$.

3.2 Security Definition

We consider a security notion for indistinguishability against key exposure and chosen plaintext attack for IKE (IND-KE-CPA). Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against IND-KE-CPA security is defined by

$$Adv_{\Pi_{IKE},\mathcal{A}}^{IND\text{-}KE\text{-}CPA}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (pp, mk) \leftarrow \mathsf{PGen}(\lambda), \\ (M_0^*, M_1^*, \mathtt{I}^*, \mathtt{time}^*, state) \leftarrow \mathcal{A}^{KG(\cdot), KI(\cdot,\cdot,\cdot)}(\mathsf{find}, pp), \\ b \xleftarrow{\$} \{0,1\}, C^* \leftarrow \mathsf{Enc}(\mathtt{I}^*, \mathtt{time}^*, M_b^*), \\ b' \leftarrow \mathcal{A}^{KG(\cdot), KI(\cdot,\cdot,\cdot)}(\mathsf{guess}, C^*, state) \end{array} \right] - \frac{1}{2} \right|.$$

where $KG(\cdot)$ and $KI(\cdot,\cdot,\cdot)$ are defined as follows.

KG$(\cdot)$**:** For a query $\mathtt{I} \in \mathcal{I}$, it stores and returns $(dk_{\mathtt{I},0}, hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)})$ by running $\mathsf{Gen}(mk, \mathtt{I})$.

KI$(\cdot,\cdot,\cdot)$**:** For a query $(i, \mathtt{I}, \mathtt{time}) \in \{0, 1, \ldots, \ell\} \times \mathcal{I} \times \mathcal{T}$, it returns $hk_{\mathtt{I},T_i(\mathtt{time})}^{(i)}$ by running $\delta_{T_{j-1}(\mathtt{time})}^{(j-1)} \leftarrow \Delta\text{-}\mathsf{Gen}(hk_{\mathtt{I},T_j(\mathtt{time})}^{(j)}, \mathtt{time})$ and $hk_{\mathtt{I},T_{j-1}(\mathtt{time})}^{(j-1)} \leftarrow \mathsf{Upd}(hk_{\mathtt{I},t}^{(j-1)}, \delta_{T_{j-1}(\mathtt{time})}^{(j-1)})$ for $j = \ell, \ldots, i+1$ (if $(dk_{\mathtt{I},0}, hk_{\mathtt{I},0}^{(1)}, \ldots, hk_{\mathtt{I},0}^{(\ell)})$ is not stored, it first generates and stores them by running Gen).

$\mathtt{I}^*$ is never issued to the KG oracle. $\mathcal{A}$ can issue any queries $(i, \mathtt{I}, \mathtt{time})$ to the KI oracle if there exists at least one *special level* $j \in \{0, 1, \ldots, \ell\}$ such that

1. For any $\mathtt{time} \in \mathcal{T}$, $(j, \mathtt{I}^*, \mathtt{time})$ is never issued to KI.
2. For any $(i, \mathtt{time}) \in \{0, 1, \ldots, j-1\} \times \mathcal{T}$ such that $T_i(\mathtt{time}) = T_i(\mathtt{time}^*)$, $(i, \mathtt{I}^*, \mathtt{time})$ is never issued to KI.

In Figure 2, we give intuition of keys that $\mathcal{A}$ can obtain by issuing to the KI oracle. In this example, let $\ell = 4$ and a special level $j = 2$.

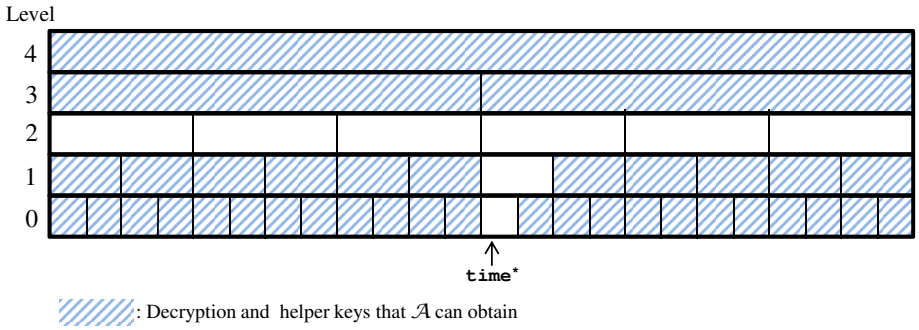

Fig. 2. Pictorial representation of secret keys for $\mathtt{I}^*$ that $\mathcal{A}$ can obtain by issuing to *KI*.

Definition 4 (IND-KE-CPA [19]). *An IKE scheme Π_{IKE} is said to be IND-KE-CPA secure if for all PPT adversaries $\mathcal{A}$, $Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda)$ is negligible in λ.*

Remark 1. As also noted in [19], there is no need to consider *key update exposure* explicitly (i.e. consider an oracle which returns any key update as much as possible) since in the above definition, $\mathcal{A}$ can get such key update from helper keys obtained from the *KI* oracle.

Remark 2. As explained in Section 1, in key-insulated cryptography including the public key setting [2, 12, 17] and the identity-based setting [19, 31, 32], two kinds of security notions have been traditionally considered: standard security and *strong* security. In most of previous works [2, 12, 17–19, 23, 31, 32], authors have considered how their scheme could achieve the strong security. We note that IND-KE-CPA security actually includes the strong security, and the fact is easily checked by setting $\ell = 1$.

By modifying the above IND-KE-CPA game so that $\mathcal{A}$ can access to the decryption oracle $Dec(\cdot,\cdot)$, which receives $(\mathtt{I}, \langle C, \mathtt{time}\rangle)$ and returns M or $\perp$, we can also define indistinguishability against key exposure and chosen ciphertext attack for IKE (IND-KE-CCA). $\mathcal{A}$ is not allowed to isuue $(\mathtt{I}\ *, \langle C^*, \mathtt{time}\rangle)$ such that $T_0(\mathtt{time}) = T_0(\mathtt{time}^*)$ to *Dec*. Let $Adv^{IND\text{-}KE\text{-}CCA}_{\Pi_{IKE},\mathcal{A}}(\lambda)$ be $\mathcal{A}$'s advantage against IND-KE-CCA security.

Definition 5 (IND-KE-CCA [19]). *An IKE scheme Π_{IKE} is said to be IND-KE-CCA secure if for all PPT adversaries $\mathcal{A}$, $Adv^{IND\text{-}KE\text{-}CCA}_{\Pi_{IKE},\mathcal{A}}(\lambda)$ is negligible in λ.*

4 Our Construction

Our basic idea is a combination of (the variant of) the Jutla–Roy HIBE [22, 25] and threshold secret sharing schemes [4, 27]. A secret B is divided into ℓ

shares $\beta_0, \ldots, \beta_{\ell-1}$, and both the secret and shares are used in exponent of a generator $g_2 \in \mathbb{G}_2$. B is embedded into the exponent of a secret key for $\mathtt{I}^*$ of the Jutla–Roy HIBE, and the resulting key is an ℓ-th level initial helper key $hk_{\mathtt{I},0}^{(\ell)}$. Roughly speaking, B works as "noise". Other initial helper keys $hk_{\mathtt{I},0}^{(i)}$ and an initial decryption key contain $g_2^{-\beta_i}$ and $g_2^{-\beta_0}$, respectively. As a lower-level key is generated, shares are eliminated from the secret B, and finally B is entirely removed when generating (or updating) a decryption key. Intuitively, since no secret keys at some special level $j \in \{0, \ldots, \ell\}$ are exposed, an adversary cannot get all β_i. Hence, he cannot generate valid decryption keys that can decrypt the challenge ciphertext for $\mathtt{I}^*$ at $\mathtt{time}^*$.

An IKE scheme Π_{IKE} =(PGen, Gen, Δ-Gen, Upd, Enc, Dec) is constructed as follows.

- $\mathsf{PGen}(\lambda, \ell)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x_0, y_0, \{(x_{1,j}, y_{1,j})\}_{j=0}^{\ell}$, $x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$z = e(g_1, g_2)^{-x_0\alpha + y_0},\ u_{1,j} := g_1^{-x_{1,j}\alpha + y_{1,j}}\ (0 \le j \le \ell),$$
$$w_1 := g_1^{-x_2\alpha + y_2},\ h_1 := g_1^{-x_3\alpha + y_3}.$$

  It outputs

$$pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{x_3}, g_2^{y_2}, g_2^{y_3}, z),$$
$$mk := (x_0, y_0).$$

- $\mathsf{Gen}(mk, ID)$: It chooses $\beta_0, \ldots, \beta_{\ell-1}, r \xleftarrow{\$} \mathbb{Z}_p$, and let $B := \sum_{i=0}^{\ell-1} \beta_i$. It computes

$$R_j := g_2^{-\beta_j}\ (0 \le j < \ell),$$
$$D_1 := (g_2^{y_2})^r,\ D_1' := g_2^{y_0}\left((g_2^{y_{1,\ell}})^{\mathtt{I}} g_2^{y_3}\right)^r,$$
$$D_2 := (g_2^{x_2})^{-r},\ D_2' := g_2^{-x_0}\left((g_2^{x_{1,\ell}})^{\mathtt{I}} g_2^{x_3}\right)^{-r},$$
$$D_3 := g_2^{r+B},$$
$$K_j := (g_2^{y_{1,j}})^r\ (0 \le j \le \ell-1),\ K_j' := (g_2^{x_{1,j}})^{-r}\ (0 \le j \le \ell-1).$$

  It outputs

$$dk_{\mathtt{I},0} := R_0,\ hk_{\mathtt{I},0}^{(i)} := R_i\ (1 \le i \le \ell-1),$$
$$hk_{\mathtt{I},0}^{(\ell)} := (D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{\ell-1}).$$

- Δ-$\mathsf{Gen}(hk_{\mathtt{I},t_i}^{(i)}, \mathtt{time})$: If $t_i \neq T_i(\mathtt{time})$, it outputs $\perp$. Otherwise, parse $hk_{\mathtt{I},t_i}^{(i)}$ as $(R_i, D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{i-1})$.[5] It chooses $\hat{r} \leftarrow \mathbb{Z}_p$, and let $t_j :=$

[5] In the case $i = \ell$, R_ℓ means an empty string, namely we have $hk_{\mathtt{I},0}^{(\ell)} := (D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{\ell-1})$.

$T_j(\mathtt{time})$ $(i-1 \le j \le \ell-1)$. It computes

$$\hat{d}_1 := D_1(g_2^{y_2})^{\hat{r}},\ \hat{d}'_1 := D'_1(K_{i-1})^{t_{i-1}}\Big((g_2^{y_{1,\ell}})^{\mathtt{I}}\prod_{j=i-1}^{\ell-1}\big((g_2^{y_{1,j}})^{t_j}\big)g_2^{y_3}\Big)^{\hat{r}},$$

$$\hat{d}_2 := D_2(g_2^{x_2})^{-\hat{r}},\ \hat{d}'_2 := D'_2(K'_{i-1})^{t_{i-1}}\Big((g_2^{x_{1,\ell}})^{\mathtt{I}}\prod_{j=i-1}^{\ell-1}\big((g_2^{x_{1,j}})^{t_j}\big)g_2^{x_3}\Big)^{-\hat{r}},$$

$$\hat{d}_3 := D_3 g_2^{\hat{r}},$$

$$\hat{k}_j := K_j(g_2^{y_{1,j}})^{\hat{r}}\ (0 \le j \le i-2),\ \hat{k}'_j := K'_j(g_2^{x_{1,j}})^{-\hat{r}}\ (0 \le j \le i-2).$$

It outputs $\delta_{t_{i-1}}^{(i-1)} := (\hat{d}_1, \hat{d}'_1, \hat{d}_2, \hat{d}'_2, \hat{d}_3, \{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{i-2})$.[6]

- $\mathsf{Upd}(hk_{\mathtt{I},t_i}^{(i)}, \delta_{\tau_i}^{(i)})$: Parse $hk_{\mathtt{I},t_i}^{(i)}$ and $\delta_{\tau_i}^{(i)}$ as $(R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1})$ and $(\hat{d}_1, \hat{d}'_1, \hat{d}_2, \hat{d}'_2, \hat{d}_3, \{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{i-1})$, respectively. It computes $D_3 := \hat{d}_3 R_i$, and sets $(D_j, D'_j) := (\hat{d}_j, \hat{d}'_j)$ $(j = 1, 2)$ and $(K_j, K'_j) := (\hat{k}_j, \hat{k}'_j)$ $(0 \le j \le i-1)$. Finally, it outputs $hk_{\mathtt{I},\tau_i}^{(i)} := (R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1})$.
- $\mathsf{Enc}(\mathtt{I}, \mathtt{time}, M)$: It chooses $s, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := Mz^s,\ C_1 := g_1^s,\ C_2 := (g_1^{\alpha})^s,\ C_3 := \Big(\prod_{j=0}^{\ell-1}\big(u_{1,j}^{t_j}\big)u_{1,\ell}^{\mathtt{I}}w_1^{\mathtt{tag}}h_1\Big)^s,$$

where $t_j := T_j(\mathtt{time})$ $(0 \le j \le \ell-1)$. It outputs $C := (C_0, C_1, C_2, C_3, \mathtt{tag})$.
- $\mathsf{Dec}(dk_{\mathtt{I},t_0}, \langle C, \mathtt{time}\rangle)$: If $t_0 \ne T_0(\mathtt{time})$, then it outputs $\perp$. Otherwise, parse $dk_{\mathtt{I},t_0}$ and C as $(R_0, D_1, D'_1, D_2, D'_2, D_3)$ and $(C_0, C_1, C_2, C_3, \mathtt{tag})$, respectively. It computes

$$M = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}}D'_1)e(C_2, D_2^{\mathtt{tag}}D'_2)}.$$

We show the correctness of our Π_{IKE}. Suppose that r denotes internal randomness of $hk_{\mathtt{I},0}^{(\ell)}$, which are generated when running $\mathsf{Gen}(mk, \mathtt{I})$, and $r^{(j)}$ denotes internal randomness of $\delta_{\mathtt{I},t_{j-1}}^{(j-1)}$ $(1 \le j \le \ell)$, which is generated when running Δ-$\mathsf{Gen}(hk_{\mathtt{I},t_j}^{(j)}, \mathtt{time})$. Then we can write $dk_{\mathtt{I},\tau_0} := (R_0, D_1, D'_1, D_2, D'_2, D_3)$ as

$$D_1 := g_2^{y_2\tilde{r}},\ D'_1 := g_2^{y_0+\tilde{r}(\mathtt{I}y_{1,\ell}+\sum_{j=0}^{\ell-1}\left(t_j y_{1,j}\right)+y_3)},$$

$$D_2 := g_2^{x_2\tilde{r}},\ D'_2 := g_2^{-x_0-\tilde{r}(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}\left(t_j x_{1,j}\right)+x_3)},\ D_3 := g_2^{\tilde{r}},$$

where $\tilde{r} := r + \sum_{i=1}^{\ell} r^{(j)}$.

[6] In the case $i = 1$, $\{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{\ell-1}$ means an empty string, namely we have $\delta_{\mathtt{I},t_0}^{(0)} := (\hat{d}_1, \ldots, \hat{d}_5)$.

Scheme	$\#pp$	$\#dk$	$\#hk_i$	$\#C$	Enc. Cost	Dec. Cost	Assumption
Ours	$(3\ell+13)\lvert\mathbb{G}\rvert$	$6\lvert\mathbb{G}\rvert$	$(2i+6)\lvert\mathbb{G}\rvert$	$4\lvert\mathbb{G}\rvert+\lvert\mathbb{Z}_p\rvert$	$[0,0,\ell+4,1]$	$[3,0,2,0]$	SXDH

Table 1. Parameters evaluation of our ℓ-level hierarchical IKE scheme. $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are cyclic groups of order p, and $|\mathbb{G}|$ denotes the bit-length of a group element in $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$, for simplicity. $|\mathcal{M}|$ and $|\mathbb{Z}_p|$ also denote the bit-length of plaintext and an element in $\mathbb{Z}_p$, respectively. $\#pp$, $\#dk$, $\#hk_i$, and $\#C$ denote sizes of public parameters, decryption keys, i-th helper keys, and ciphertexts, respectively. In computational cost analysis, $[\cdot,\cdot,\cdot,\cdot]$ means the number of [pairing, multi-exponentiation, regular exponentiation, fixed-based exponentiation]. For comparison we mention that relative tunings for the various operations are as follows: [pairing≈ 5, multi-exp≈ 1.5, regular-exp:$= 1$, fixed-based-exp$\ll 0.2$].

Suppose that $dk_{\mathtt{I},t_0} = (R_0, D_1, D_1', D_2, D_2', D_3)$ and $C = (C_0, C_1, C_2, C_3, \mathtt{tag})$ are correctly generated. Then, we have

$$\begin{aligned}
&\frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} D_1') e(C_2, D_2^{\mathtt{tag}} D_2')} \\
&= M e(g_1, g_2)^{(-x_0\alpha + y_0)s} \\
&\cdot \frac{e(g_1^{s(\sum_{j=0}^{\ell-1} t_j(-x_{1,j}\alpha + y_{1,j}) + \mathtt{I}(-x_{1,\ell}\alpha + y_{1,\ell}) + \mathtt{tag}(-x_2\alpha + y_2) - x_3\alpha + y_3)}, g_2^{\tilde{r}})}{e(g_1^s, g_2^{y_2\tilde{r}\mathtt{tag} + y_0 + \tilde{r}(\mathtt{I}y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j y_{1,j}) + y_3)}) e(g_1^{\alpha s}, g_2^{-x_2\tilde{r}\mathtt{tag} - x_0 - \tilde{r}(\mathtt{I}x_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j x_{1,j}) + x_3)})} \\
&= M e(g_1, g_2)^{(-x_0\alpha + y_0)s} \frac{1}{e(g_1^s, g_2^{y_0}) e(g_1^{\alpha s}, g_2^{-x_0})} = M.
\end{aligned}$$

We obtain the following theorem. The proof is postponed to Section 5.

Theorem 1. *If the SXDH assumption holds, then the resulting ℓ-level hierarchical IKE scheme Π_{IKE} is IND-KE-CPA secure.*

4.1 Parameters Evaluation and Comparison

First, we show the parameter sizes and computational costs of our hierarchical IKE scheme in Table 1.

Also, an efficiency comparison between our IKE scheme and the existing IKE schemes [19, 32] is given in Table 2. In fact, the WLC+08 scheme [32] has the threshold property and does not have a hierarchical structure, and therefore, we set the threshold value is one in the WLC+08 scheme and the hierarchy depth is one in the HHSI05 scheme [19] and our scheme for the fair comparison. The HHSI05 scheme meets the IND-KE-CCA security, however the scheme is secure only in the random oracle model (ROM). Both the WLC+08 scheme and ours meet the IND-KE-CPA security in the standard model (i.e. without random oracles). Although assumptions behind these schemes (i.e. the computational

Scheme	$\#pp$	$\#dk$	$\#hk$	$\#C$	Enc. Cost	Dec. Cost	Assumption
HHSI05 [19] ($\ell = 1$)	$2\|\mathbb{G}\|$	$3\|\mathbb{G}\|$	$\|\mathbb{G}\|$	$3\|\mathbb{G}\| + \|\mathcal{M}\|$	$[1, 0, 2, 1]$	$[4, 0, 2, 1]$	CBDH (in ROM)
WLC+08 [32]	$(2n + 5)\|\mathbb{G}\|$	$4\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$4\|\mathbb{G}\|$	$[0, 1, 3, 1]$	$[3, 0, 0, 0]$	DBDH
Ours ($\ell = 1$)	$16\|\mathbb{G}\|$	$6\|\mathbb{G}\|$	$7\|\mathbb{G}\|$	$4\|\mathbb{G}\| + \|\mathbb{Z}_p\|$	$[0, 0, 5, 1]$	$[3, 0, 2, 0]$	SXDH

Table 2. Efficiency comparison between our construction and existing schemes. The notation used here is the same as that in Table 1 except for $\#hk$, which denotes the helper key size. What n appears in public-parameter sizes means that the public-parameter size depends on the size of its identity space.

bilinear Diffie–Hellman (CBDH), decisional bilinear Diffie–Hellman (DBDH),[7] and SXDH assumptions) are different, they all are static and simple. We emphasize that the threshold structure does not strengthen the underlying DBDH assumption of the WLC+08 scheme since the structure was realized via only threshold secret sharing techniques [4, 27]. Note that we do not take into account the parallel IKE scheme [31] since the model of the scheme is slightly different from those of the above schemes. However, the public parameter size of the parallel IKE scheme also depends on the size of its identity space, and we mention that this is due to the underlying Waters IBE [29], not due to the parallel property.

As can be seen, we first achieve the IKE scheme with constant-size parameters in the standard model. Again, we also get the first IKE scheme in the hierarchical setting without random oracles.

5 Proof of Security

We describe how semi-functional ciphertexts and secret keys are generated as follows.

Semi-functional Ciphertext: Parse a normal ciphertext C as $(C_0, C_1, C_2, C_3, \mathtt{tag})$. A semi-functional ciphertext $\widetilde{C} := (\tilde{C}_0, \tilde{C}_1, \tilde{C}_2, \tilde{C}_3, \widetilde{\mathtt{tag}})$ is computed as follows:

$$\tilde{C}_0 := C_0 e(g_1, g_2)^{-x_0\mu} = M e(g_1, g_2)^{-x_0(\alpha s+\mu)+y_0 s},$$
$$\tilde{C}_1 := C_1,$$
$$\tilde{C}_2 := C_2 g_1^{\mu} = g_1^{\alpha s+\mu},$$
$$\tilde{C}_3 := C_3 \Big((g_1^{x_{1,\ell}})^{\mathtt{I}} \prod_{j=0}^{\ell-1} \big((g_1^{x_{1,j}})^{t_j} \big) (g_1^{x_2})^{\mathtt{tag}} g_1^{x_3} \Big)^{-\mu}$$

[7] The formal definitions of the CBDH and DBDH assumptions are given in Appendix A.

$$= C_3 g_1^{-\mu(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j x_{1,j})+x_2\mathtt{tag}+x_3)}$$
$$= g_1^{-(\alpha s+\mu)(\mathtt{I}x_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j x_{1,j})+x_2\mathtt{tag}+x_3)} g_1^{s(\mathtt{I}y_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j y_{1,j})+y_2\mathtt{tag}+y_3)},$$

and $\widetilde{\mathtt{tag}} := \mathtt{tag}$, where $\mu \xleftarrow{\$} \mathbb{Z}_p$.

Semi-functional Decryption and Helper Key: Parse a normal helper key $hk_{\mathtt{I},t_i}^{(i)}$ as $(R_i, D_1, D_1', D_2, D_2', D_3, \{(K_j, K_j')\}_{j=0}^{i-1})$. A semi-functional helper key $\widetilde{hk}_{\mathtt{I},t_i}^{(i)} := (\tilde{R}_i, \tilde{D}_1, \tilde{D}_1', \tilde{D}_2, \tilde{D}_2', \tilde{D}_3, \{(\tilde{K}_j, \tilde{K}_j')\}_{j=0}^{i-1})$ is computed as follows: $R_i := \tilde{R}_i$,

$$\begin{aligned}
\tilde{D}_1 &:= D_1 g_2^{\gamma} = g_2^{y_2 r+\gamma},\\
\tilde{D}_1' &:= D_1 g_2^{\gamma\phi} = g_2^{y_0+r(\mathtt{I}y_{1,\ell}+\sum_{j=i}^{\ell-1}(t_j y_{1,j})+y_3)+\gamma\phi},\\
\tilde{D}_2 &:= D_2 g_2^{-\frac{\gamma}{\alpha}} = g_2^{-rx_2-\frac{\gamma}{\alpha}},\\
\tilde{D}_2' &:= D_2 g_2^{-\frac{\gamma\phi}{\alpha}} = g_2^{-x_0-r(\mathtt{I}x_{1,\ell}+\sum_{j=i}^{\ell-1}(t_j x_{1,j})+x_3)-\frac{\gamma\phi}{\alpha}},\\
\tilde{D}_3 &:= D_3,\\
\tilde{K}_j &:= K_j g_2^{\gamma\phi_j} = g_2^{ry_{1,j}+\gamma\phi_j} \ (0 \le j \le i-1),\\
\tilde{K}_j' &:= K_j' g_2^{-\frac{\gamma\phi_j}{\alpha}} = g_2^{-rx_{1,j}-\frac{\gamma\phi_j}{\alpha}} \ (0 \le j \le i-1),
\end{aligned}$$

where $\gamma, \phi, \{\phi_j\}_{j=0}^{i-1} \xleftarrow{\$} \mathbb{Z}_p$. Note that $hk_{\mathtt{I},t_0}^{(0)}$ means $dk_{\mathtt{I},t_0}$ for any $t_0 \in \mathcal{T}_0$. In particular, $\widetilde{hk}_{\mathtt{I},t_0}^{(0)}$ $(= \widetilde{dk}_{\mathtt{I},t_0})$ is called a semi-functional decryption key. We also note that in order to generate the semi-functional decryption or helper key, $g_2^{\frac{1}{\alpha}}$ is needed in addition to the public parameter.

A semi-functional ciphertext can be decrypted with a normal key. This fact can be easily checked by

$$\frac{e(g_1^{\mu(\mathtt{I}y_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j y_{1,j})+y_2\mathtt{tag}+y_3)}, D_3) e(g_1, g_2)^{-x_0\mu}}{e(g_1^{\mu}, D_1^{\mathtt{tag}} D_1')} = 1.$$

Also, a normal ciphertext can be decrypted with a semi-functional decryption key since it holds $e(C_1, g_2^{\gamma\mathtt{tag}} g_2^{\gamma\phi}) e(C_2, g_2^{-\frac{\gamma}{\alpha}\mathtt{tag}} g_2^{-\frac{\gamma\phi}{\alpha}}) = 1$.

A helper or decryption key obtained by running the Δ-Gen and Upd algorithms with a semi-functional helper key is also semi-functional.

Proof (of Theorem 1). Based on [22, 25], we prove the theorem through a sequence of games. We first define the following games:

$\mathsf{Game}_{\mathsf{Real}}$: This is the same as the IND-KE-CPA game described in Section 3.

Game_0: This is the same as $\mathsf{Game}_{\mathsf{Real}}$ except that the challenge ciphertext is semi-functional.

Game_k $(1 \le k \le q)$**:** This is the same as Game_0 except for the following modification: Let q be the maximum number of identities issued to the KG or KI oracles, and $\mathtt{I}_i$ $(1 \le i \le q)$ be an i-th identity issued to the oracles. If queries regarding the first k identities $\mathtt{I}_1, \ldots, \mathtt{I}_k$ are issued, then semi-functional decryption and/or helper keys are returned. The rest of keys (i.e., keys regarding $\mathtt{I}_{k+1}, \ldots, \mathtt{I}_q$) are normal.

$\mathsf{Game}_{\mathsf{Final}}$: This is the same as Game_q except that the challenge ciphertext is a semi-functional one of a random element of $\mathbb{G}_T$.

Let S_{Real}, S_k $(0 \le k \le q)$, and S_{Final} be the probabilities that the event $b' = b$ occurs in $\mathsf{Game}_{\mathsf{Real}}$, Game_k, and $\mathsf{Game}_{\mathsf{Final}}$, respectively. Then, we have

$$Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda) \le |S_{\mathsf{Real}} - S_0| + \sum_{i=1}^{q} |S_{i-1} - S_i| + |S_q - S_{\mathsf{Final}}| + |S_{\mathsf{Final}} - \frac{1}{2}|.$$

The rest of the proof follows from the following lemmas.

Lemma 1. *If the DDH1 assumption holds, then it holds that* $|S_{\mathsf{Real}} - S_0| \le Adv^{DDH1}_{\mathcal{G},\mathcal{B}}(\lambda)$.

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_1^{c_1}, g_1^{c_2}, g_2, T)$ of the DDH1 problem. Then, $\mathcal{B}$ randomly chooses $x_0, y_0, \{(x_{1,j}, y_{1,j})\}_{j=0}^{\ell}, x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$, and creates

$$z := e(g_1^{c_1}, g_2)^{-x_0} e(g_1, g_2)^{y_0},\ u_{1,j} := (g_1^{c_1})^{-x_{1,j}} g_1^{y_{1,j}}\ (0 \le j \le \ell),$$
$$w_1 := (g_1^{c_1})^{-x_2} g_1^{y_2},\ h_1 := (g_1^{c_1})^{-x_3} g_1^{y_3}.$$

$\mathcal{B}$ sends $pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{x_3}, g_2^{y_2}, g_2^{y_3}, z)$ to $\mathcal{A}$. Note that $\mathcal{B}$ knows a master key $mk := (x_0, y_0)$ and we implicitly set $\alpha := c_1$.

$\mathcal{B}$ can simulate the KG and KI oracles since $\mathcal{B}$ knows the master key.

In the challenge phase, $\mathcal{B}$ receives $(M_0^*, M_1^*, \mathtt{I}^*, \mathtt{time}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$. $\mathcal{B}$ chooses $\mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$, and let $t_j^* := T_j(\mathtt{time}^*)$ $(0 \le j \le \ell - 1)$. $\mathcal{B}$ computes

$$C_0^* := M_d e(T, g_2)^{-x_0} e(g_1^{c_2}, g_2)^{y_0},\ C_1^* := g_1^{c_2},\ C_2^* := T,$$
$$C_3^* := T^{-\mathtt{I}^* x_{1,\ell} - \sum_{j=0}^{\ell-1}(t_j^* x_{1,j}) - x_2 \mathtt{tag}^* - x_3} (g_1^{c_2})^{\mathtt{I}^* y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* y_{1,j}) + y_2 \mathtt{tag}^* + y_3}.$$

$\mathcal{B}$ sends $C^* := (C_0^*, C_1^*, C_2^*, C_3^*, \mathtt{tag}^*)$ to $\mathcal{A}$.

If $b = 0$, then the above ciphertext is normal by setting $s := c_2$. If $b = 1$, then the above ciphertext is semi-functional since it holds

$$C_0^* = M_d e(g_1, g_2)^{-x_0(c_1 c_2 + \mu) + y_0 c_2} = M_d e(g_1, g_2)^{-x_0(\alpha s + \mu) + y_0 s},$$
$$C_2^* = g_1^{c_1 c_2 + \mu} = g_1^{\alpha s + \mu},$$
$$C_3^* = g^{-(c_1 c_2 + \mu)(\mathtt{I}^* x_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* x_{1,j}) + x_2 \mathtt{tag}^* + x_3)} g_1^{c_2(\mathtt{I}^* y_{1,\ell} + \sum_{j=0}^{\ell-1}(t_j^* y_{1,j}) + y_2 \mathtt{tag}^* + y_3)}$$

$$= g^{-(\alpha s+\mu)(\mathtt{I}^* x_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j^* x_{1,j})+x_2\mathtt{tag}^*+x_3)} g_1^{s(\mathtt{I}^* y_{1,\ell}+\sum_{j=0}^{\ell-1}(t_j^* y_{1,j})+y_2\mathtt{tag}^*+y_3)}.$$

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the DDH1 problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

Lemma 2. *For every $k \in \{1, \ldots, q\}$, if the DDH2 assumption holds, then it holds that $|S_{k-1} - S_k| \leq Adv^{DDH2}_{\mathcal{G},\mathcal{B}}(\lambda)$.*

Proof. At the beginning, a PPT adversary $\mathcal{B}$ receives an instance $(g_1, g_2, g_2^{c_1}, g_2^{c_2}, T)$ of the DDH2 problem. Then, $\mathcal{B}$ randomly chooses $x'_0, y_0, \{(x'_{1,j}, y'_{1,j}, y''_{1,j})\}_{j=0}^{\ell}, x'_2, x'_3$, $y'_3, y''_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^\times$, and (implicitly) sets

$$\begin{aligned}
&x_0 := \frac{x'_0 + y_0}{\alpha},\ x_{1,j} := \frac{x'_{1,j} + y_{1,j}}{\alpha},\ \text{where } y_{1,j} := y'_{1,j} + c_2 y''_{1,j}\ (0 \leq j \leq \ell),\\
&x_2 := \frac{x'_2 + c_2}{\alpha},\ y_2 := c_2,\\
&x_3 := \frac{x'_3 + y_3}{\alpha},\ \text{where } y_3 := y'_3 + c_2 y''_3.
\end{aligned}$$

$\mathcal{B}$ creates

$$\begin{aligned}
&z := e(g_1, g_2)^{-x'_0},\ u_{1,j} := g_1^{-x'_{1,j}}\ (0 \leq j \leq \ell),\ w_1 := g_1^{-x'_2},\ h_1 := g_1^{-x'_3},\\
&g_2^{x_{1,j}} := g_2^{\frac{x'_{1,j}+y'_{1,j}}{\alpha}} (g_2^{c_2})^{\frac{y''_{1,j}}{\alpha}}\ (0 \leq j \leq \ell),\ g_2^{y_{1,j}} := g_2^{y'_{1,j}} (g_2^{c_2})^{y''_{1,j}}\ (0 \leq j \leq \ell),\\
&g_2^{x_2} := g_2^{\frac{x'_2}{\alpha}} (g_2^{c_2})^{\frac{1}{\alpha}},\ g_2^{y_2} := g_2^{c_2},\ g_2^{x_3} := g_2^{\frac{x'_3+y'_3}{\alpha}} (g_2^{c_2})^{\frac{y''_3}{\alpha}},\ g_2^{y_3} := g_2^{y'_3} (g_2^{c_2})^{y''_3}.
\end{aligned}$$

$\mathcal{B}$ sends $pp := (g_1, g_1^\alpha, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{y_2}, g_2^{x_3}, g_2^{y_3}, z)$ to $\mathcal{A}$. Note that $\mathcal{B}$ knows a master key $mk := (x_0, y_0)$.

We show how $\mathcal{B}$ simulates the *KG* and *KI* oracles. Let $\mathtt{I}_i$ $(1 \leq i \leq q)$ be an i-th identity issued to the oracles. Without loss of generality, we consider $\mathcal{A}$ issues all identities $\mathtt{I}_i \neq \mathtt{I}^*$ to the *KG* oracle, and issues only queries regarding $\mathtt{I}^*$ to the *KI* oracle.

KG oracle. $\mathcal{B}$ creates $k-1$ semi-functional decryption and helper keys, and embeds T into the k-th keys. The rest of keys are normal.

Case $i < k$**:** After receiving $\mathtt{I}_i$, $\mathcal{B}$ creates and returns semi-functional keys. Since $\mathcal{B}$ knows the master key and α, $\mathcal{B}$ can create both normal and semi-functional keys.

Case $i = k$**:** After receiving $\mathtt{I}_k$, $\mathcal{B}$ creates semi functional keys by embedding T as follows: $\mathcal{B}$ chooses $\beta_0, \ldots, \beta_{\ell-1} \xleftarrow{\$} \mathbb{Z}_p$ and sets $B := \sum_{j=0}^{\ell-1} \beta_j$. $\mathcal{B}$ computes

$$\begin{aligned}
&R_j := g_2^{-\beta_j}\ (0 \leq j < \ell),\\
&D_1 := T,
\end{aligned}$$

$$
\begin{aligned}
D'_1 &:= g_2^{y_0}(g_2^{c_1})^{\mathtt{I}_k y'_{1,\ell}+y'_3} T^{\mathtt{I}_k y''_{1,\ell}+y''_3},\\
D_2 &:= \left((g_2^{c_1})^{x'_2} T\right)^{-\frac{1}{\alpha}},\\
D'_2 &:= g_2^{-\frac{x'_0}{\alpha}}(g_2^{c_1})^{-\frac{\mathtt{I}_k(x'_{1,\ell}+y'_{1,\ell})+x'_3+y'_3}{\alpha}} g_2^{-\frac{y_0}{\alpha}} T^{-\frac{\mathtt{I}_k y''_{1,\ell}+y''_3}{\alpha}},\\
D_3 &:= g_2^{c_1} g_2^{B},\\
K_j &:= (g_2^{c_1})^{y'_{1,j}}(T)^{y''_{1,j}} \ (0 \le j \le \ell-1),\\
K'_j &:= (g_2^{c_1})^{-\frac{x'_{1,j}+y'_{1,j}}{\alpha}} T^{-\frac{y''_{1,j}}{\alpha}} \ (0 \le j \le \ell-1).
\end{aligned}
$$

$\mathcal{B}$ sets $dk_{\mathtt{I},0} := R_0$, $hk^{(i)}_{\mathtt{I},0} := R_i$ $(1 \le i \le \ell-1)$, $hk^{(\ell)}_{\mathtt{I},0} := (D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{\ell-1})$. If $b = 0$, then it is easy to see that the above keys are normal by setting $r := c_1$. If $b = 1$, then the above ciphertext is semi-functional since it holds

$$
\begin{aligned}
D_1 :=& T = g_2^{c_1 c_2+\gamma} = g_2^{y_2 r+\gamma},\\
D'_1 :=& g_2^{y_0}(g_2^{c_1})^{\mathtt{I}_k y'_{1,\ell}+y'_3} T^{\mathtt{I}_k y''_{1,\ell}+y''_3}\\
=& g_2^{y_0+c_1(\mathtt{I}_k(y'_{1,\ell}+c_2 y''_{1,\ell})+y'_3+c_2 y''_3)} g_2^{\gamma(\mathtt{I}_k y''_{1,\ell}+y''_3)} = g_2^{y_0+r(\mathtt{I}_k y_{1,\ell}+y_3)} g_2^{\gamma\phi},\\
D_2 :=& \left((g_2^{c_1})^{x'_2} T\right)^{-\frac{1}{\alpha}} = g_2^{-\frac{c_1(x'_2+c_2)}{\alpha}} g_2^{-\frac{\gamma}{\alpha}} = g_2^{-r x_2} g_2^{-\frac{\gamma}{\alpha}},\\
D'_2 :=& g_2^{-\frac{x'_0}{\alpha}}(g_2^{c_1})^{-\frac{\mathtt{I}_k(x'_{1,\ell}+y'_{1,\ell})+x'_3+y'_3}{\alpha}} g_2^{-\frac{y_0}{\alpha}} T^{-\frac{\mathtt{I}_k y''_{1,\ell}+y''_3}{\alpha}}\\
=& g_2^{-\frac{(x'_0+y_0)+c_1(\mathtt{I}_k(x'_{1,\ell}+y'_{1,\ell}+c_2 y''_{1,\ell})+(x'_3+y'_3+c_2 y''_3))}{\alpha}} g_2^{-\frac{\gamma(\mathtt{I}_k y''_{1,\ell}+y''_3)}{\alpha}}\\
=& g_2^{-x_0-r(\mathtt{I}_k x_{1,\ell}+x_3)} g_2^{-\frac{\gamma\phi}{\alpha}},\\
K_j :=& (g_2^{c_1})^{y'_{1,j}}(T)^{y''_{1,j}} = g_2^{c_1(y'_{1,j}+c_2 y''_{1,j})} g_2^{\gamma y''_{1,j}} = g_2^{r y_{1,j}} g_2^{\gamma\phi_j} \ (0 \le j \le \ell-1),\\
K'_j :=& (g_2^{c_1})^{-\frac{x'_{1,j}+y'_{1,j}}{\alpha}} T^{-\frac{y''_{1,j}}{\alpha}}\\
=& g_2^{-\frac{c_1(x'_{1,j}+y'_{1,j}+c_2 y''_{1,j})}{\alpha}} g_2^{-\frac{\gamma y''_{1,j}}{\alpha}} = g_2^{-r x_{1,j}} g_2^{-\frac{\gamma\phi_j}{\alpha}} \ (0 \le j \le \ell-1),
\end{aligned}
$$

where $T := g_2^{c_1 c_2+\gamma}$, $r := c_1$, $\phi := \mathtt{I}_k y''_{1,\ell} + y''_3$, and $\phi_j := y''_{1,j}$ $(0 \le j \le \ell-1)$. Since $y''_{1,j}$ and y''_3 are chosen uniformly at random, ϕ and ϕ_j are also uniformly distributed.

Case $i > k$**:** After receiving $\mathtt{I}_i$, $\mathcal{B}$ creates and returns normal keys by using the master key.

KI oracle. Suppose that $\mathcal{A}$ issues $k-1$ identities $\mathtt{I}_1, \ldots, \mathtt{I}_{k-1}$ to the *KG* oracle, and then issues a query $(i, \mathtt{I}^*, \mathtt{time})$ (i.e., $\mathtt{I}^*(= \mathtt{I}_k)$) to the *KI* oracle. Note that for some special level $j \in \{0, \ldots, \ell\}$, $\mathcal{A}$ cannot issue $\mathtt{time}$ such that $T_i(\mathtt{time}) = T_i(\mathtt{time}^*)$ if $i < j$ ($\mathcal{B}$ does not need to know where level is special one in advance). $\mathcal{B}$ creates and stores semi-functional decryption and helper keys

$(\widetilde{d}_{\mathtt{I}^*,0}, \widetilde{hk}^{(1)}_{\mathtt{I}^*,0}, \ldots, \widetilde{hk}^{(\ell)}_{\mathtt{I}^*,0})$ as in the case $i = k$ of the KG oracle. We also note that from the second query, $\mathcal{B}$ answers queries by using the stored keys. Then, $\mathcal{B}$ repeatedly runs $\delta^{(j-1)}_{t_{j-1}} \leftarrow \Delta\text{-}\mathsf{Gen}(hk^{(j)}_{\mathtt{I}^*,t_j}, \mathtt{time}^*)$ and $hk^{(j-1)}_{\mathtt{I}^*,t^*_{j-1}} \mathsf{Upd}(hk^{(j-1)}_{\mathtt{I}^*,0}, \delta^{(j-1)}_{t_{j-1}})$ for $j = \ell, \ldots, i+1$, where $t_\ell := 0$ and $t_j := T_j(\mathtt{time})$ $(0 \leq j \leq \ell - 1)$. Again, the key generated by semi-functional helper keys is also semi-functional. $\mathcal{B}$ returns $hk^{(i)}_{\mathtt{I}^*,t_i}$ to $\mathcal{A}$.

In the challenge phase, $\mathcal{B}$ receives $(M^*_0, M^*_1, \mathtt{I}^*, \mathtt{time}^*)$ from $\mathcal{A}$. $\mathcal{B}$ chooses $d \xleftarrow{\$} \{0,1\}$, and sets $t^*_j := T_j(\mathtt{time}^*)$ $(0 \leq j \leq \ell - 1)$. However, $\mathcal{B}$ cannot create the semi-functional ciphertext for $\mathtt{I}^*$ without knowledge of c_2 (and hence $y_{1,j}$ $(0 \leq j \leq \ell)$ and y_3). To generate the semi-functional ciphertext without the knowledge, $\mathcal{B}$ sets

$$\widetilde{\mathtt{tag}}^* := -\sum_{j=0}^{\ell-1}(t^*_j y''_{1,j}) - \mathtt{I}^* y''_{1,\ell} - y''_3.$$

Since $y''_{1,0}, \ldots, y''_{1,\ell}$ and y''_3 are chosen uniformly at random, probability distribution of $\widetilde{\mathtt{tag}}^*$ is also uniformly at random from $\mathcal{A}$'s view.[8] Then, $\mathcal{B}$ chooses $s, \mu \xleftarrow{\$} \mathbb{Z}_p$, and computes

$$\begin{aligned}
\tilde{C}^*_0 :=& M^*_d z^s e(g_1, g_2)^{-x_0\mu} = M^*_d e(g_1, g_2)^{-x_0(\alpha s+\mu)+y_0 s},\\
\tilde{C}^*_1 :=& g_1^s,\\
\tilde{C}^*_2 :=& g_1^{\alpha s+\mu}\\
\tilde{C}^*_3 :=& \Big(\prod_{j=0}^{\ell-1}(u^{t^*_j}_{1,j}) u^{\mathtt{I}}_{1,\ell} w_1^{\widetilde{\mathtt{tag}}^*} h_1\Big)^s g_1^{\mu(y'_3+\sum_{j=0}^{\ell-1}(t^*_j y'_{1,j})+\mathtt{I}^* y'_{1,\ell})}\\
=& \Big(\prod_{j=0}^{\ell-1}(u^{t^*_j}_{1,j}) u^{\mathtt{I}}_{1,\ell} w_1^{\widetilde{\mathtt{tag}}^*} h_1\Big)^s\\
& \cdot g_1^{\mu(\sum_{j=0}^{\ell-1}(t^*_j(y'_{1,j}+c_2 y''_{1,j}))+\mathtt{I}^*(y'_{1,\ell}+c_2 y''_{1,\ell})+c_2\widetilde{\mathtt{tag}}^*+y'_3+c_2 y''_3)}\\
& \cdot g_1^{-c_2\mu(\sum_{j=0}^{\ell-1}(t^*_j y''_{1,j})+\mathtt{I}^* y''_{1,\ell}+\widetilde{\mathtt{tag}}^*+y''_3)}\\
=& \Big(\prod_{j=0}^{\ell-1}(u^{t^*_j}_{1,j}) u^{\mathtt{I}}_{1,\ell} w_1^{\widetilde{\mathtt{tag}}^*} h_1\Big)^s g_1^{\mu(\sum_{j=0}^{\ell-1}(t^*_j y_{1,j})+\mathtt{I}^* y_{1,\ell}+y_2\widetilde{\mathtt{tag}}^*+y_3)}.
\end{aligned}$$

$\mathcal{B}$ sends $\widetilde{C}^* := (\tilde{C}^*_0, \tilde{C}^*_1, \tilde{C}^*_2, \tilde{C}^*_3, \widetilde{\mathtt{tag}}^*)$ to $\mathcal{A}$.

After receiving d' from $\mathcal{A}$, $\mathcal{B}$ sends $b' = 1$ to the challenger of the DDH2 problem if $d' = d$. Otherwise, $\mathcal{B}$ sends $b' = 0$ to the challenger. □

[8] The fact that the formula in such a form is uniformly distributed was traditionally studied in the context of unconditionally secure authentication protocols (e.g., [5, 21, 28]).

Lemma 3. $|S_q - S_{\mathsf{Final}}| \leq \frac{q}{p}$.

Proof. We modify the setup procedure and the semi-functional keys generation procedure in Game_q, and the modification turns out $\mathsf{Game}_{\mathsf{Final}}$. We show that before and after the modification are statistically indistinguishable without probability $\frac{q}{p}$.

In the setup phase, we randomly choose $x'_0, y_0, \{(x'_{1,j}, y_{1,j})\}_{j=0}^{\ell}, x'_2, y_2, x'_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and set

$$x_0 := \frac{x'_0 + y_0}{\alpha},\ x_{1,j} := \frac{x'_{1,j} + y_{1,j}}{\alpha}\ (0 \leq j \leq \ell),\ x_2 := \frac{x'_2 + y_2}{\alpha},\ x_3 := \frac{x'_3 + y_3}{\alpha}.$$

$\mathcal{B}$ creates

$$z := e(g_1, g_2)^{-x'_0},\ u_{1,j} := g_1^{-x'_{1,j}}\ (0 \leq j \leq \ell),\ w_1 := g_1^{-x'_2},\ h_1 := g_1^{-x'_3},$$
$$g_2^{x_{1,j}} := g_2^{\frac{x'_{1,j}+y_{1,j}}{\alpha}}\ (0 \leq j \leq \ell),\ g_2^{x_2} := g_2^{\frac{x'_2+y_2}{\alpha}},\ g_2^{x_3} := g_2^{\frac{x'_3+y_3}{\alpha}}.$$

We set $pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{x_2}, g_2^{y_2}, g_2^{x_3}, g_2^{y_3}, z)$ and $mk := (x_0, y_0)$.

When generating (initial) semi-functional keys, we choose $\beta_0, \ldots, \beta_{\ell-1}, r, \phi', \phi'_0, \ldots, \phi'_{\ell-1}, \gamma \xleftarrow{\$} \mathbb{Z}_p$, and (implicitly) set $B := \sum_{j=0}^{\ell-1} \beta_i$, $\phi' := y_0 + r(\mathtt{I} y_{1,\ell} + y_3) + \gamma\phi$, and $\phi'_j := r y_{1,j} + \gamma\phi_j$ $(0 \leq j \leq \ell - 1)$. We compute

$$\begin{aligned}
\tilde{R}_j :=& g_2^{-\beta_j}\ (0 \leq j \leq \ell - 1),\\
\tilde{D}_1 :=& g_2^{y_2 r + \gamma},\\
\tilde{D}'_1 :=& g_2^{\phi'} = g_2^{y_0 + r(\mathtt{I} y_{1,\ell} + y_3) + \gamma\phi},\\
\tilde{D}_2 :=& g_2^{-r\frac{x'_2+y_2}{\alpha} - \frac{\gamma}{\alpha}} = g_2^{-r x_2 - \frac{\gamma}{\alpha}},\\
\tilde{D}'_2 :=& g_2^{-\frac{1}{\alpha}(\phi' + x'_0 + r(x'_3 + \mathtt{I} x'_{1,\ell}))}\\
=& g_2^{-\frac{1}{\alpha}(x'_0 + y_0 + r\mathtt{I}(x'_{1,\ell} + y_{1,\ell}) + \gamma\phi + r(x'_3 + y_3))} = g_2^{-x_0 - r(\mathtt{I} x_{1,\ell} + x_3) - \frac{\gamma\phi}{\alpha}},\\
\tilde{D}_3 :=& g_2^{r+B},\\
\tilde{K}_j :=& g_2^{\phi'_j} = g_2^{r y_{1,j} + \gamma\phi_j}\ (0 \leq j \leq \ell - 1),\\
\tilde{K}'_j :=& g_2^{-\frac{r x'_{1,j} + \phi'_j}{\alpha}} = g_2^{-\frac{r(x'_{1,j} + y_{1,j}) + \gamma\phi_j}{\alpha}} = g_2^{-r x_{1,j} - \frac{\gamma\phi_j}{\alpha}}\ (0 \leq j \leq \ell - 1).
\end{aligned}$$

We set $dk_{\mathtt{I},0} := \tilde{R}_0$, $hk_{\mathtt{I},0}^{(j)} := \tilde{R}_j$ $(1 \leq j \leq \ell - 1)$, and $hk_{\mathtt{I},0}^{(\ell)} := (\tilde{D}_1, \tilde{D}'_1, \tilde{D}_2, \tilde{D}'_2, \tilde{D}_3, \{(\tilde{K}_j, \tilde{K}'_j)\}_{j=0}^{\ell-1})$. We emphasize that although the above secret keys are well-formed, y_0, $\{y_{1,j}\}_{j=0}^{\ell}$, and y_3 are not used in the above procedure.

On the other hand, the first component of the challenge ciphertext is generated as $\tilde{C}_0^* := M_b z^s e(g_1, g_2)^{-x_0\mu} = M_b e(g_1, g_2)^{-x_0(\alpha s + \mu) + y_0}$. This means that y_0, which is independent of secret keys and public parameters, masks $\tilde{C}_0^*$, and hence $\tilde{C}_0^*$ becomes the ciphertext of a random element of $\mathbb{G}_T$.

Since γ is chosen uniformly at random, ϕ and ϕ_j are distributed uniformly at random if $\gamma \neq 0$. An event that $\gamma = 0$ occurs with probability $1/p$. Every query regarding $\mathtt{I}_i$ $(1 \leq i \leq q)$ may cause this event, and hence, we have $|S_q - S_{\mathsf{Final}}| \leq \frac{q}{p}$. □

Proof of Theorem 1. From Lemmas 1, 2, and 3, we have $Adv^{IND\text{-}KE\text{-}CPA}_{\Pi_{IKE},\mathcal{A}}(\lambda) \leq |S_{\mathsf{Real}} - S_0| + \sum_{i=1}^{q} |S_{i-1} - S_i| + |S_q - S_{\mathsf{Final}}| + |S_{\mathsf{Final}} - \frac{1}{2}| \leq Adv^{DDH1}_{\mathcal{G},\mathcal{B}}(\lambda) + q \cdot Adv^{DDH2}_{\mathcal{G},\mathcal{B}}(\lambda) + \frac{q}{p}$. □

6 Chosen-Ciphertext Security

Boneh et al. [6] proposed an well-known transformation from $\ell + 1$-level CPA-secure HIBE (and one-time signature (OTS)) to ℓ-level CCA-secure HIBE. We cannot apply this transformation to a hierarchical IKE scheme *in a generic way* since it does not have delegating functionality. However, we can apply their techniques to the underlying Jutla–Roy HIBE of our hierarchical IKE, and therefore we obtain CCA-secure scheme. We show the detailed construction as follows. We assume a verification key vk is appropriately encoded as an element of $\mathbb{Z}_p$ when it is used in exponent of ciphertexts.

Let $\Pi_{OTS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ be an OTS scheme.[9] An ℓ-level hierarchical IKE scheme $\Pi_{IKE} =$(PGen, Gen, Δ-Gen, Upd, Enc, Dec) is constructed as follows.

- $\mathsf{PGen}(\lambda, \ell)$: It runs $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, g_1, g_2, e) \leftarrow \mathcal{G}$. It chooses $x_0, y_0, \{(x_{1,j}, y_{1,j})\}_{j=0}^{\ell}$, $\hat{x}_1, \hat{y}_1, x_2, y_2, x_3, y_3 \xleftarrow{\$} \mathbb{Z}_p$ and $\alpha \xleftarrow{\$} \mathbb{Z}_p^{\times}$, and sets

$$z = e(g_1, g_2)^{-x_0\alpha + y_0},\ u_{1,j} := g_1^{-x_{1,j}\alpha + y_{1,j}}\ (0 \leq j \leq \ell),$$
$$\hat{u}_1 := g_1^{-\hat{x}_1\alpha + \hat{y}_1},\ w_1 := g_1^{-x_2\alpha + y_2},\ h_1 := g_1^{-x_3\alpha + y_3}.$$

  It outputs

$$pp := (g_1, g_1^{\alpha}, \{u_{1,j}\}_{j=0}^{\ell}, \hat{u}_1, w_1, h_1, g_2, \{(g_2^{x_{1,j}}, g_2^{y_{1,j}})\}_{j=0}^{\ell}, g_2^{\hat{x}_1}, g_2^{\hat{y}_1}, g_2^{x_2}, g_2^{x_3}, g_2^{y_2}, g_2^{y_3}, z),$$
$$msk := (x_0, y_0).$$

- $\mathsf{Gen}(mk, ID)$: It chooses $\beta_0, \ldots, \beta_{\ell-1}, r \xleftarrow{\$} \mathbb{Z}_p$, and let $B := \sum_{i=0}^{\ell-1} \beta_i$. It computes

$$R_j := g_2^{-\beta_j}\ (0 \leq j < \ell),$$
$$D_1 := (g_2^{y_2})^r,\ D_1' := g_2^{y_0}\left((g_2^{y_{1,\ell}})^{\mathtt{I}} g_2^{y_3}\right)^r,$$
$$D_2 := (g_2^{x_2})^{-r},\ D_2' := g_2^{-x_0}\left((g_2^{x_{1,\ell}})^{\mathtt{I}} g_2^{x_3}\right)^{-r},$$

[9] The formal description of the OTS is given in Appendix A.

$$D_3 := g_2^{r+B},$$
$$K_j := (g_2^{y_{1,j}})^r \ (0 \le j \le \ell-1),\ K'_j := (g_2^{x_{1,j}})^{-r} \ (0 \le j \le \ell-1),$$
$$K_{vk} := (g_2^{\hat{y}_1})^r,\ K'_{vk} := (g_2^{\hat{x}_1})^{-r}.$$

It outputs

$$dk_{\mathrm{I},0} := R_0,\ hk_{\mathrm{I},0}^{(i)} := R_i \ (1 \le i \le \ell-1),$$
$$hk_{\mathrm{I},0}^{(\ell)} := (D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{\ell-1}, K_{vk}, K'_{vk}).$$

- Δ-$\mathsf{Gen}(hk_{\mathrm{I},t_i}^{(i)}, \mathtt{time})$: If $t_i \neq T_i(\mathtt{time})$, it outputs $\perp$. Otherwise, parse $hk_{\mathrm{I},t_i}^{(i)}$ as $(R_i, D_1, D'_1,\ D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1}, K_{vk}, K'_{vk})$. It chooses $\hat{r} \leftarrow \mathbb{Z}_p$, and let $t_j := T_j(\mathtt{time})$ $(i-1 \le j \le \ell-1)$. It computes

$$\hat{d}_1 := D_1(g_2^{y_2})^{\hat{r}},\ \hat{d}'_1 := D'_1(K_{i-1})^{t_{i-1}}\Big((g_2^{y_{1,\ell}})^{\mathrm{I}} \prod_{j=i-1}^{\ell-1}\big((g_2^{y_{1,j}})^{t_j}\big)g_2^{y_3}\Big)^{\hat{r}},$$
$$\hat{d}_2 := D_2(g_2^{x_2})^{-\hat{r}},\ \hat{d}'_2 := D'_2(K'_{i-1})^{t_{i-1}}\Big((g_2^{x_{1,\ell}})^{\mathrm{I}} \prod_{j=i-1}^{\ell-1}\big((g_2^{x_{1,j}})^{t_j}\big)g_2^{x_3}\Big)^{-\hat{r}},$$
$$\hat{d}_3 := D_3 g_2^{\hat{r}},$$
$$\hat{k}_j := K_j(g_2^{y_{1,j}})^{\hat{r}} \ (0 \le j \le i-2),\ \hat{k}'_j := K'_j(g_2^{x_{1,j}})^{-\hat{r}} \ (0 \le j \le i-2),$$
$$\hat{k}_{vk} := K_{vk}(g_2^{\hat{y}_1})^{\hat{r}},\ \hat{k}'_{vk} := K'_{vk}(g_2^{\hat{x}_1})^{\hat{r}}.$$

It outputs $\delta_{t_{i-1}}^{(i-1)} := (\hat{d}_1, \hat{d}'_1, \hat{d}_2, \hat{d}'_2, \hat{d}_3, \{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{i-2}, \hat{k}_{vk}, \hat{k}'_{vk})$.

- $\mathsf{Upd}(hk_{\mathrm{I},t_i}^{(i)}, \delta_{\tau_i}^{(i)})$: Parse $hk_{\mathrm{I},t_i}^{(i)}$ and $\delta_{\tau_i}^{(i)}$ as $(R_i, D_1, D'_1, D_2, D'_2, D_3, \{(K_j, K'_j)\}_{j=0}^{i-1}$, $K_{vk}, K'_{vk})$ and $(\hat{d}_1, \hat{d}'_1, \hat{d}_2, \hat{d}'_2, \hat{d}_3, \{(\hat{k}_j, \hat{k}'_j)\}_{j=0}^{i-1}, \hat{k}_{vk}, \hat{k}_{vk})$, respectively. It computes $D_3 := \hat{d}_3 R_i$, and sets $(D_j, D'_j) := (\hat{d}_j, \hat{d}'_j)$ $(j = 1, 2)$, $(K_j, K'_j) := (\hat{k}_j, \hat{k}'_j)$ $(0 \le j \le i-1)$, and $(K_{vk}, K'_{vk}) := (\hat{k}_{vk}, \hat{k}'_{vk})$. Finally, it outputs $hk_{\mathrm{I},\tau_i}^{(i)} := (R_i, D_1, D'_1, D_2, D'_2,\ D_3, \{(K_j, K'_j)\}_{j=0}^{i-1}, K_{vk}, K'_{vk})$.
- $\mathsf{Enc}(\mathtt{I}, \mathtt{time}, M)$: It first runs $(vk, sk) \leftarrow \mathsf{KGen}(\lambda)$. It chooses $s, \mathtt{tag} \xleftarrow{\$} \mathbb{Z}_p$. For $M \in \mathbb{G}_T$, it computes

$$C_0 := Mz^s,\ C_1 := g_1^s,\ C_2 := (g_1^{\alpha})^s,\ C_3 := \Big(\prod_{j=0}^{\ell-1}(u_{1,j}^{t_j})u_{1,\ell}^{\mathrm{I}}\hat{u}_1^{vk}w_1^{\mathtt{tag}}h_1\Big)^s,$$

where $t_j := T_j(\mathtt{time})$ $(0 \le j \le \ell-1)$. It also runs $\sigma \leftarrow \mathsf{Sign}(sk, (C_0, C_1, C_2, C_3, \mathtt{tag}))$, and outputs $C := (vk, C_0, C_1, C_2, C_3, \mathtt{tag}, \sigma)$.
- $\mathsf{Dec}(dk_{\mathrm{I},t_0}, \langle C, \mathtt{time}\rangle)$: If $t_0 \neq T_0(\mathtt{time})$, then it outputs $\perp$. Otherwise, parse dk_{I,t_0} and C as $(R_0, D_1, D'_1, D_2, D'_2, D_3, K_{vk}, K'_{vk})$ and $(vk, C_0, C_1, C_2, C_3, \mathtt{tag}, \sigma)$, respectively. If $\mathsf{Ver}(vk, C_0, C_1, C_2, C_3, \mathtt{tag}, \sigma) \to 0$, then it outputs $\perp$. Otherwise, it computes

$$\hat{D}'_1 := D'_1(K_{vk})^{vk},\ \hat{D}'_2 := D'_2(K'_{vk})^{vk}.$$

Finally, it outputs

$$M = \frac{C_0 e(C_3, D_3)}{e(C_1, D_1^{\mathtt{tag}} \hat{D}'_1) e(C_2, D_2^{\mathtt{tag}} \hat{D}'_2)}.$$

The correctness of the above IKE scheme Π_{IKE} can be checked as in our CPA-secure IKE scheme described in Section 4.

We obtain the following theorem. The proof is omitted since this theorem can be easily proved by combining Boneh et al.'s techniques [6] and our proof techniques of Theorem 1.

Theorem 2. *If the underlying OTS scheme Π_{OTS} is sUF-OT secure and the SXDH assumption holds, then the resulting ℓ-level hierarchical IKE scheme Π_{IKE} is IND-KE-CCA secure.*

7 Conclusion

In this paper, we first proposed hierarchical IKE scheme in the standard model. When the hierarchy is one, our scheme achieves constant-size parameters including public parameters, decryption and helper keys, and ciphertexts, and hence our scheme is more efficient than the existing scheme [32] in the sense of parameter sizes. Our scheme is based on the Jutla–Roy HIBE [22] (and its variant [25]) and techniques of threshold secret sharing schemes [4, 27].

Acknowledgments. We would like to thank anonymous PKC 2016 referees for their helpful comments. The first author is supported by JSPS Research Fellowships for Young Scientists. This work (Yohei Watanabe) was supported by Grant-in-Aid for JSPS Fellows Grant Number 25·3998. This work (Junji Shikata) was partially conducted under the auspices of the MEXT Program for Promoting the Reform of National Universities.

References

1. Bellare, M., Miner, S.: A forward-secure digital signature scheme. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 431–448. Springer Berlin Heidelberg (1999)
2. Bellare, M., Palacio, A.: Protecting against key-exposure: strongly key-insulated encryption with optimal threshold. Applicable Algebra in Engineering, Communication and Computing 16(6), 379–396 (2006)
3. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. pp. 321–334. S&P'07 (May 2007)
4. Blakley, G.: Safeguarding cryptographic keys. In: Proceedings of the 1979 AFIPS National Computer Conference. pp. 313–317. AFIPS Press, Monval, NJ, USA (1979)

5. den Boer, B.: A simple and key-economical unconditional authentication scheme. Journal of Computer Security 2, 65–72 (1993)
6. Boneh, D., Canetti, R., Halevi, S., Katz, J.: Chosen ciphertext security from identity based encryption. SIAM Journal on Computing 36(5), 1301–1328 (2007)
7. Boneh, D., Sahai, A., Waters, B.: Functional encryption: Definitions and challenges. In: Ishai, Y. (ed.) Theory of Cryptography. Lecture Notes in Computer Science, vol. 6597, pp. 253–273. Springer Berlin Heidelberg (2011)
8. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) Advances in Cryptology — EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 255–271. Springer Berlin Heidelberg (2003)
9. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings — the role of Ψ revisited. Discrete Applied Mathematics 159(13), 1311 – 1322 (2011)
10. Dodis, Y., Franklin, M., Katz, J., Miyaji, A., Yung, M.: Intrusion-resilient public-key encryption. In: Joye, M. (ed.) Topics in Cryptology — CT-RSA 2003. Lecture Notes in Computer Science, vol. 2612, pp. 19–32. Springer Berlin Heidelberg (2003)
11. Dodis, Y., Franklin, M., Katz, J., Miyaji, A., Yung, M.: A generic construction for intrusion-resilient public-key encryption. In: Okamoto, T. (ed.) Topics in Cryptology — CT-RSA 2004. Lecture Notes in Computer Science, vol. 2964, pp. 81–98. Springer Berlin Heidelberg (2004)
12. Dodis, Y., Katz, J., Xu, S., Yung, M.: Key-insulated public key cryptosystems. In: Knudsen, L. (ed.) Advances in Cryptology EUROCRYPT 2002. Lecture Notes in Computer Science, vol. 2332, pp. 65–82. Springer Berlin Heidelberg (2002)
13. Dodis, Y., Katz, J., Xu, S., Yung, M.: Strong key-insulated signature schemes. In: Desmedt, Y. (ed.) Public Key Cryptography — PKC 2003. Lecture Notes in Computer Science, vol. 2567, pp. 130–144. Springer Berlin Heidelberg (2002)
14. Dodis, Y., Luo, W., Xu, S., Yung, M.: Key-insulated symmetric key cryptography and mitigating attacks against cryptographic cloud software. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. pp. 57–58. ASIACCS '12, ACM, New York, NY, USA (2012)
15. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113 – 3121 (2008)
16. Gentry, C., Silverberg, A.: Hierarchical ID-based cryptography. In: Zheng, Y. (ed.) Advances in Cryptology - ASIACRYPT 2002. Lecture Notes in Computer Science, vol. 2501, pp. 548–566. Springer Berlin Heidelberg (2002)
17. Hanaoka, G., Hanaoka, Y., Imai, H.: Parallel key-insulated public key encryption. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds.) Public Key Cryptography — PKC 2006. Lecture Notes in Computer Science, vol. 3958, pp. 105–122. Springer Berlin Heidelberg (2006)
18. Hanaoka, G., Weng, J.: Generic constructions of parallel key-insulated encryption. In: Garay, J., De Prisco, R. (eds.) Security and Cryptography for Networks. vol. 6280, pp. 36–53. Springer Berlin Heidelberg (2010)
19. Hanaoka, Y., Hanaoka, G., Shikata, J., Imai, H.: Identity-based hierarchical strongly key-insulated encryption and its application. In: Roy, B. (ed.) Advances in cryptology — ASIACRYPT 2005. Lecture Notes in Computer Science, vol. 3788, pp. 495–514. Springer Berlin Heidelberg (2005)
20. Itkis, G., Reyzin, L.: SiBIR: Signer-base intrusion-resilient signatures. In: Yung, M. (ed.) Advances in Cryptology — CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 499–514. Springer Berlin Heidelberg (2002)

21. Johansson, T., Kabatianskii, G., Smeets, B.: On the relation between A-codes and codes correcting independent errors. In: Helleseth, T. (ed.) Advances in Cryptology — EUROCRYPT'93. Lecture Notes in Computer Science, vol. 765, pp. 1–11. Springer Berlin Heidelberg (1994)
22. Jutla, C., Roy, A.: Shorter quasi-adaptive NIZK proofs for linear subspaces. In: Sako, K., Sarkar, P. (eds.) Advances in Cryptology — ASIACRYPT 2013. Lecture Notes in Computer Science, vol. 8269, pp. 1–20. Springer Berlin Heidelberg (2013)
23. Libert, B., Quisquater, J.J., Yung, M.: Parallel key-insulated public key encryption without random oracles. In: Okamoto, T., Wang, X. (eds.) Public Key Cryptography — PKC 2007. Lecture Notes in Computer Science, vol. 4450, pp. 298–314. Springer Berlin Heidelberg (2007)
24. Ramanna, S., Chatterjee, S., Sarkar, P.: Variants of Waters' dual system primitives using asymmetric pairings. In: Fischlin, M., Buchmann, J., Manulis, M. (eds.) Public Key Cryptography — PKC 2012. Lecture Notes in Computer Science, vol. 7293, pp. 298–315. Springer Berlin Heidelberg (2012)
25. Ramanna, S., Sarkar, P.: Efficient (anonymous) compact HIBE from standard assumptions. In: Chow, S., Liu, J., Hui, L., Yiu, S. (eds.) Provable Security. Lecture Notes in Computer Science, vol. 8782, pp. 243–258. Springer International Publishing (2014)
26. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) Advances in Cryptology —EUROCRYPT 2005. Lecture Notes in Computer Science, vol. 3494, pp. 457–473. Springer Berlin Heidelberg (2005)
27. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (Nov 1979)
28. Taylor, R.: An integrity check value algorithm for stream ciphers. In: Stinson, D. (ed.) Advances in Cryptology — CRYPTO'93. Lecture Notes in Computer Science, vol. 773, pp. 40–48. Springer Berlin Heidelberg (1994)
29. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) Advances in Cryptology – EUROCRYPT 2005. vol. 3494, pp. 114–127. Springer Berlin Heidelberg (2005)
30. Waters, B.: Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In: Halevi, S. (ed.) Advances in Cryptology — CRYPTO 2009. vol. 5677, pp. 619–636. Springer Berlin Heidelberg (2009)
31. Weng, J., Liu, S., Chen, K., Ma, C.: Identity-based parallel key-insulated encryption without random oracles: Security notions and construction. In: Barua, R., Lange, T. (eds.) Progress in Cryptology - INDOCRYPT 2006. vol. 4329, pp. 409–423. Springer Berlin Heidelberg (2006)
32. Weng, J., Liu, S., Chen, K., Zheng, D., Qiu, W.: Identity-based threshold key-insulated encryption without random oracles. In: Malkin, T. (ed.) Topics in Cryptology – CT-RSA 2008. vol. 4964, pp. 203–220. Springer Berlin Heidelberg (2008)

A Definitions

We give the formal definitions of the CBDH and DBDH assumptions and OTS. In the following, we assume the Type-1 pairing (i.e., $\mathbb{G} := \mathbb{G}_1 = \mathbb{G}_2$).

Computational Bilinear Diffie–Hellman (CBDH) Assumption. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the CBDH problem

as follows.

$$Adv^{CBDH}_{\mathcal{G},\mathcal{A}}(\lambda) := \Pr\left[T = e(g,g)^{c_1c_2c_3} \;\middle|\; \begin{array}{l} (p,\mathbb{G},\mathbb{G}_T,g,e) \leftarrow \mathcal{G}, \\ c_1,c_2,c_3 \xleftarrow{\$} \mathbb{Z}_p, \\ T \leftarrow \mathcal{A}(\lambda,g,g^{c_1},g^{c_2},g^{c_3}) \end{array} \right].$$

Definition 6. *The CBDH assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv^{CBDH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.*

Decisional Bilinear Diffie–Hellman (DBDH) Assumption. Let $\mathcal{A}$ be a PPT adversary and we consider $\mathcal{A}$'s advantage against the DBDH problem as follows.

$$Adv^{DBDH}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| \Pr\left[b' = b \;\middle|\; \begin{array}{l} (p,\mathbb{G},\mathbb{G}_T,g,e) \leftarrow \mathcal{G}, \\ c_1,c_2,c_3 \xleftarrow{\$} \mathbb{Z}_p, \\ b \xleftarrow{\$} \{0,1\}, \\ \text{if } b = 1 \text{ then } W := \hat{e}(g,g)^{c_1c_2c_3}, \\ \text{else } W \xleftarrow{\$} \mathbb{G}_T, \\ b' \leftarrow \mathcal{A}(\lambda,g,g^{c_1},g^{c_2},g^{c_3},W) \end{array} \right] - \frac{1}{2} \right|.$$

Definition 7. *The DBDH assumption relative to a generator $\mathcal{G}$ holds if for all PPT adversaries $\mathcal{A}$, $Adv^{DBDH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is negligible in λ.*

One-time signature. An OTS scheme Π_{OTS} consists of three-tuple algorithms (KGen, Sign, Ver) defined as follows.

- $(vk, sk) \leftarrow \mathsf{KGen}(\lambda)$: It takes a security parameter λ and outputs a pair of a public key and a secret key (vk, sk).
- $\sigma \leftarrow \mathsf{Sign}(sk, m)$: It takes the secret key sk and a message $m \in \mathcal{M}$ and outputs a signature σ.
- 1 or $0 \leftarrow \mathsf{Ver}(vk, m, \sigma)$: It takes the public key vk and a pair of a message and a signature (m, σ), and then outputs 1 or 0.

We assume that Π_{OTS} meets the following *correctness* property: For all $\lambda \in \mathbb{N}$, all $(vk, sk) \leftarrow \mathsf{KGen}(\lambda)$, and all $m \in \mathcal{M}$, it holds that $1 \leftarrow \mathsf{Ver}(vk, (m, \mathsf{Sign}(sk, m)))$.

We describe the notion of strong unforgeability against one-time attack ($\mathsf{sUF\text{-}OT}$). Let $\mathcal{A}$ be a PPT adversary, and $\mathcal{A}$'s advantage against sUF-OT security is defined by

$$Adv^{sUF\text{-}OT}_{\Pi_{OTS},\mathcal{A}}(\lambda) := \Pr\left[1 \leftarrow \mathsf{Ver}(vk, m^*, \sigma^*) \wedge (m^*, \sigma^*) \neq (m, \sigma) \;\middle|\; \begin{array}{l} (vk, sk) \leftarrow \mathsf{KGen}(\lambda), \\ (m^*, \sigma^*) \leftarrow \mathcal{A}^{Sign(\cdot)}(vk) \end{array} \right].$$

Sign$(\cdot)$ is a *signing oracle* which takes a message m as input, and then returns σ by running $\mathsf{Sign}(sk, m)$. $\mathcal{A}$ is allowed to access to the above oracle only once.

Definition 8. *An OTS scheme Π_{OTS} is said to be sUF-OT secure if for all PPT adversaries $\mathcal{A}$, $Adv^{sUF\text{-}OT}_{\Pi_{OTS},\mathcal{A}}(\lambda)$ is negligible in λ.*