Generic Constructions of Parallel Key-Insulated Encryption: Stronger Security Model and Novel Schemes*

Goichiro Hanaoka[1], Jian Weng[2]

[1]National Institute of Advanced Industrial Science and Technology, Tokyo, Japan
[3]Department of Computer Science, Jinan University, Guangzhou 510632, P.R. China

Abstract

Exposure of a secret key is a significant threat in practice. As a notion of security against key exposure, Dodis et al. advocated key-insulated security, and proposed concrete key-insulated encryption (KIE) schemes in which secret keys are periodically updated by using a physically "insulated" helper key. For significantly reducing possibility of exposure of the helper key, Hanaoka et al. further proposed the notion of parallel KIE (PKIE) in which multiple helper keys are used in alternate shifts. They also pointed out that in contrast to the case of the standard KIE, PKIE cannot be straightforwardly obtained from identity-based encryption (IBE). In this paper, we first discuss that previous security models for PKIE are somewhat weak, and thus re-formalize stronger security models for PKIE. Then we clarify that PKIE can be generically constructed (even in the strenghthened security models) by using a new primitive which we call one-time forward secure public key encryption (OTFS-PKE) and show that it is possible to construct OTFS-PKE from arbitrary IBE or hierarchical IBE (without degenerating into IBE). By using our method, we can obtain various new PKIE schemes which yield desirable properties. For example, we can construct first PKIE schemes from lattice or quadratic residuosity problems (without using bilinear maps), and PKIE with short ciphertexts and cheaper computational cost for both encryption and decryption. Interestingly, the resulting schemes can be viewed as the partial solutions to the open problem left by Libert, Quisquarter and Yung in PKC'07.

Keywords: key exposure, parallel key-insulated encryption, one-time forward secure public key encryption, identity-based encryption.

1 Introduction

Background. Nowadays, there is a growing tendency for cryptographic systems to be deployed on inexpensive, lightweight and mobile devices. In such a situation, a secret key is more casually and frequently used, and thus, both damage and possibility of key exposure increase significantly.

Key-insulated cryptography, introduced by Dodis, Katz, Xu and Yung [21], is a useful technique to mitigate the potential damage caused by key exposure. In a key-insulated encryption (KIE) scheme, the lifetime of the system is divided into discrete periods, and the pubic key remains fixed throughout the lifetime. Each user keeps two kinds of secrets which are called *user secret key* and *helper key*. The user secret key is used for decrypting ciphertexts, and the helper key is used for updating the user secret key. Since the user secret key is periodically updated (without changing the public key), its exposure compromises only security during its corresponding time period, and security of other time periods (including past ones) are still maintained. Furthermore, the helper key is stored in a dedicated device named *helper*, which is kept isolated from the network except when it is used for updating the user secret key, and thus we can assume that possibility of its exposure is very low.

*A preliminary version of this paper will be presented at *Seventh Conference on Security and Cryptography for Networks (SCN 2010) [28].*

The key-insulation paradigm is an effective solution to the key exposure problem. However, we also notice that it is not easy to simultaneously handle (1) reducing damage by exposure of the user secret key, and (2) reducing possibility of exposure of the helper key. Namely, if we update the user secret key more frequently, then exposure of the user secret key compromises security for only a shorter time period. But, this also increases frequency of the helper's connection to insecure environments, and hence increases the risk of helper key exposure. We note that exposure of the helper key may compromise security of *all* time periods.

To address the above problem, Hanaoka, Hanaoka, and Imai [26] introduced the concept of parallel key-insulated encryption (PKIE), where two (or more) distinct helpers are *alternately* used to update the user secret keys. As indicated in [26], PKIE allows frequent updating of the user secret key, and at the same time reduces the risk of the helper key exposure. Namely, assuming that in some time period, the user secret key is updated by using one of two helper keys, next updating procedure cannot be carried out without using the other helper key. Therefore, even if one of helper keys is exposed, its damage is still very limited.

Based on Boneh-Franklin identity-based encryption (IBE) scheme [8], Hanaoka et al. [26] also proposed a concrete PKIE scheme (referred to as HHI scheme) in the random oracle model [5]. Later, based on Boneh-Boyen IBE scheme [6], Libert, Quisquater and Yung [36] further proposed another PKIE scheme (referred to as LQY scheme) without using random oracles.

Libert et al. also pointed out an important fact that in contrast to the standard KIE, *it is not straightforward to construct PKIE from IBE*. Actually, it has not been known if it is possible to generically construct PKIE from any IBE or not, and therefore, we cannot flexibly design PKIE schemes according to individual system requirement.

Our Results. In this paper, we first discuss that previous security models for PKIE are somewhat weak, and thus re-formalize stronger security models for PKIE. Next, we show that it is possible to generically construct a PKIE scheme (even in the strengthened security models) from an arbitrary IBE scheme, and give various useful instantiations. Specifically, we first introduce a new primitive named one-time forward secure public key encryption (OTFS-PKE), and then present a generic construction of PKIE from OTFS-PKE. Furthermore, we present two generic constructions of OTFS-PKE: one is from standard IBE, and the other is from two-level hierarchical identity-based encryption (2-HIBE). We note that IBE can be trivially obtained from 2-HIBE, but our generalization based on OTFS-PKE yields more flexibility which results in a wider range of applications.

First examples of instantiations of our generic construction are PKIE schemes from various assumptions. Namely, by converting lattice-based IBE schemes [16, 24, 37] into OTSF-PKE schemes, we immediately have PKIE schemes based on difficulty of lattice problems. These schemes can be considered as the first "post-quantum" PKIE schemes. Similarly, based on the quadratic-residuosity-based IBE schemes [9, 17], we can easily construct PKIE schemes from the same underlying assumptions. These are the first PKIE schemes from a factoring-related problem. We stress that all previously known PKIE schemes depend on pairings, and thus the above schemes can also be the first PKIE schemes without pairings.

Second examples are PKIE schemes with better efficiency in comparison to existing schemes (e.g. [36]) in some aspects. For instance, based on Boneh-Boyen IBE [6], we can construct a new PKIE scheme with shorter ciphertexts and cheaper computational cost for encryption and decryption. For another instance, based on Boneh-Boyen-Goh HIBE [7], we can also construct a new PKIE scheme which yields cheaper computation for helpers. This scheme is useful when helpers are computationally weak devices (e.g. smart cards). Surprisingly, when our Boneh-Boyen-based scheme is extended to support multiple helpers, its public key size, ciphertext size, encryption cost and decryption cost are all constant and independent with the number of helpers. This scheme can be viewed as a (partial) solution to the open question left by Libert et al. in PKC'07.

Related Works. In their seminal paper, Dodis, Katz, Xu and Yung [21] presented generic constructions as well as direct constructions of KIE schemes. Bellare and Palacio [4] proposed a new

KIE scheme based on Boneh-Franklin IBE scheme, and they also presented the generic construction of (non-strong key-insulated) KIE from IBE. We remark that these generic constructions cannot be applied to PKIE systems. Hanaoka et al. [27] studied KIE in the unconditional settings, and proposed a dynamic and mutual KIE scheme.

Phan et al. [38] generalized the notion of PKIE and introduced a new paradigm called key-insulated public key encryption with auxiliary helper. Weng et al. [43] extended PKIE to identity-based scenarios, and proposed an identity-based PKIE scheme without random oracles. However, the efficiency of their scheme also degrades with the number of helpers. Hanaoka et al. [29] introduced the paradigm of hierarchial key-insulation, and presented the constructions of identity-based hierarchial KIE. Weng et al. [44] introduced the notion of threshold key-insulation, and proposed an identity-based threshold KIE scheme.

There exist some other related techniques to deal with the key exposure problem. Forward secure cryptography [1, 3, 12, 14, 32, 35] can ensure that, exposure of the current key does not render usages of previous keys insecure, but security of the future periods is lost. Intrusion-resilient cryptography [18, 19, 31] strengths the key-insulated security in the sense that, the system remains secure even after arbitrarily many compromises of both helper key and user secret keys, as long as the compromises are not simultaneous.

2 Preliminaries

2.1 Notations

For a bit $\mathsf{flg} \in \{0,1\}$, $\overline{\mathsf{flg}}$ denotes $1 - \mathsf{flg}$. For a finite set S, $x \xleftarrow{\$} \mathcal{S}$ means choosing an element x from S with a uniform distribution. For a string x, $|x|$ denotes its bit-length. We use $\mathcal{A}(x, y, \cdots)$ to denote that $\mathcal{A}$ is an algorithm with the input $(x, y, \cdots)$, and use $z \leftarrow \mathcal{A}(x, y, \cdots)$ to denote the running of $\mathcal{A}(x, y, \cdots)$ with the output z. For $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$, it means that $\mathcal{A}$ is an algorithm with the input $(x, y, \cdots)$ and can access to oracles $\mathcal{O}_1, \mathcal{O}_2, \cdots$. By $z \leftarrow \mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$, we denote the running of $\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2, \cdots}(x, y, \cdots)$ with the output z. Throughout this paper, we use $\mathcal{M}$ to denote the message space of the encryption schemes. A function $F : \mathbb{N} \to [0, 1]$ is said to be *negligible* if for all $c \in \mathbb{N}$ there exists a $k_c \in \mathbb{N}$ such that $F(k) < k^{-c}$ for all $k > k_c$.

2.2 Public Key Encryption

A public key encryption (PKE) scheme $\mathtt{PKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three algorithms: The *key generation* algorithm $(pk, sk) \leftarrow \mathsf{KGen}(\lambda)$, taking as input a security parameter λ, outputs a public/secret key pair (pk, sk). The *encryption* algorithm $C \leftarrow \mathsf{Enc}(pk, M)$, on input a public key pk and a message $M \in \mathcal{M}$, outputs a ciphertext C. The *decryption* algorithm $M \leftarrow \mathsf{Dec}(sk, C)$, on input a cipertext C and the secret key sk, outputs a plaintext M (or "$\perp$" if C is invalid).

Next, we review the definition of semantic security [25] for PKE, i.e. $\mathsf{IND\text{-}ATK}$ [2, 22, 40] where $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$. Let $\mathcal{B}$ be an adversary running in two stages `find` and `guess`. Consider the following experiment:

$$\mathbf{Exp}^{\mathsf{IND\text{-}ATK}}_{\mathcal{B},\mathtt{PKE}}(\lambda)\text{:}\big[(pk, sk) \leftarrow \mathsf{KeyGen}(\lambda);\ (M_0, M_1) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathrm{d}}(\cdot)}_{\mathtt{find}}(pk);\ \delta \xleftarrow{\$} \{0,1\};$$
$$C^* \leftarrow \mathsf{Enc}(pk, M_\delta);\ \delta' \leftarrow \mathcal{B}^{\mathcal{O}_{\mathrm{d}}(\cdot)}_{\mathtt{guess}}(C^*, pk);\ \text{return 1 if } \delta = \delta', \text{ or 0 otherwise}\ \big],$$

where $\mathcal{O}_{\mathrm{d}}(\cdot)$ is a decryption oracle which for given C returns $M \leftarrow \mathsf{Dec}(sk, C)$ if $\mathsf{ATK} = \mathsf{CCA}$, or "$\perp$" if $\mathsf{ATK} = \mathsf{CPA}$. It is required that $|M_0| = |M_1|$, and $\mathcal{B}$ cannot issue the query $\mathcal{O}_{\mathrm{d}}(C^*)$. We define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathsf{IND\text{-}ATK}}_{\mathcal{B},\mathtt{PKE}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}ATK}}_{\mathcal{B},\mathtt{PKE}}(\lambda) = 1] - \frac{1}{2}\right|$.

Definition 1 *We say that a* PKE *scheme is* IND-CCA *(resp.* IND-CPA*) secure, if there exists no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ who has advantage* $\mathrm{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathcal{B},\mathtt{PKE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{B},\mathtt{PKE}}(\lambda)$*).*

2.3 Hierarchical Identity-Based Encryption

Hierarchical identity-based encryption (HIBE) is a generalization of IBE that mirrors an organizational hierarchy. An HIBE scheme $\mathtt{HIBE} = (\mathsf{Setup}, \mathsf{Extract}, \mathsf{Enc}, \mathsf{Dec})$ consists of four algorithms: The *setup* algorithm $(param, msk) \leftarrow \mathsf{Setup}(\lambda, l)$, run by the private key generator (PKG), on input a security parameter λ and the maximum hierarchy depth l, outputs the public parameters $param$ and the master secret key msk. The *key extraction* algorithm $sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(param, sk_{\mathsf{ID}_{|k-1}}, \mathsf{ID})$, on input $param$, an identity $\mathsf{ID} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_k)$ of depth $k \leq l$, and the secret key $sk_{\mathsf{ID}_{|k-1}}$ of the parent identity $\mathsf{ID}_{|k-1} = (\mathsf{ID}_1, \cdots, \mathsf{ID}_{k-1})$, outputs the secret key sk_{ID} for ID. The *encryption* algorithm $C \leftarrow \mathsf{Enc}(param, \mathsf{ID}, M)$, on input $param$, a message $M \in \mathcal{M}$ and an identity ID with a depth equal or less than l, outputs a ciphertext C. The *decryption* algorithm $m \leftarrow \mathsf{Dec}(sk_{\mathsf{ID}}, C)$, on input a secret key sk_{ID} and a ciphertext C, outputs a plaintext M (or "$\perp$" if C is invalid).

Note that when the maximal hierarchy depth in HIBE degenerates to a single level, it yields an IBE system. For IBE, we use $\mathsf{Setup}(\lambda)$ instead of $\mathsf{Setup}(\lambda, l)$ to denote its setup algorithm. Next, we review the definition of semantic security for HIBE/IBE, i.e. IND-ID-ATK where $\mathsf{ATK} = \{\mathsf{CPA}, \mathsf{CCA}\}$. For an adversary $\mathcal{A}$, we consider the following experiment:

$$\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}ATK}}_{\mathcal{A},\mathtt{HIBE/IBE}}(\lambda)\colon \big[(param, msk) \leftarrow \mathsf{Setup}(\lambda, l);\ (M_0, M_1, \mathsf{ID}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{ext}}(\cdot),\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{find}}(param);\ \delta \xleftarrow{\$} \{0,1\};$$
$$C^* \leftarrow \mathsf{Enc}(param, \mathsf{ID}^*, M_\delta);\ \delta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{ext}}(\cdot),\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{guess}}(param, C^*);\ \text{return 1 if } \delta = \delta' \text{ or 0 otherwise}\big],$$

where $\mathcal{O}_{\mathrm{ext}}(\cdot)$ is a key extraction oracle which for given ID returns $sk_{\mathsf{ID}} \leftarrow \mathsf{Extract}(param, msk, \mathsf{ID})$, and $\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)$ is a decryption oracle which for given (ID, C) returns $M \leftarrow \mathsf{Dec}(sk_{\mathsf{ID}}, C)$ if $\mathsf{ATK} = \mathsf{CCA}$, or "$\perp$" if $\mathsf{ATK} = \mathsf{CPA}$. It is required that $|M_0| = |M_1|$, $\mathcal{A}$ cannot submit ID^* nor a prefix of ID^* to oracle $\mathcal{O}_{\mathrm{ext}}$, and $\mathcal{A}$ cannot submit (ID^*, C^*) to oracle $\mathcal{O}_{\mathrm{d}}$. We define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathsf{IND\text{-}ID\text{-}ATK}}_{\mathcal{A},\mathtt{HIBE/IBE}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}ATK}}_{\mathcal{A},\mathtt{HIBE/IBE}}(\lambda) = 1] - \frac{1}{2}\right|$.

Definition 2 *An* HIBE/IBE *scheme is said to be* IND-ID-CCA *(resp.* IND-ID-CPA*) secure, if there exits no PPT adversary $\mathcal{A}$ who has non-negligible advantage* $\mathrm{Adv}^{\mathsf{IND\text{-}ID\text{-}CCA}}_{\mathcal{A},\mathtt{HIBE/IBE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{IND\text{-}ID\text{-}CPA}}_{\mathcal{A},\mathtt{HIBE/IBE}}(\lambda)$*).*

Remark 1. Canetti, Halevi, and Katz [14] defined a weaker notion of security for HIBE/IBE (i.e., IND-sID-CCA, IND-sID-CPA), in which the adversary must declare the identity ID^* he intends to attack before seeing the public parameters of the system.

2.4 Parallel Key-Insulated Encryption

In this subsection, we first review the definition of PKIE systems as defined in [26, 36], with slight notational differences. Then, we further re-formalize the security model for PKIE which is stronger than previous one [26, 36].

Formally, a PKIE scheme consists of the following algorithms:

- $\mathsf{KeyGen}(\lambda)$: The *key generation* algorithm, on input a security parameter λ, outputs a public key pk, an initial user secret key usk_0 and two helper keys $(\mathsf{mst}_1, \mathsf{mst}_0)$. Here usk_0 is kept by the user, while mst_1 and mst_0 are kept by the first and the second helper respectively. We write $(\mathsf{pk}, \mathsf{usk}_0, (\mathsf{mst}_1, \mathsf{mst}_0)) \leftarrow \mathsf{KeyGen}(\lambda)$.
- $\Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \mod 2})$: The *helper key-update* algorithm is run by the helpers at the beginning of each period. On input a period index t and the corresponding helper key $\mathsf{mst}_{t \mod 2}$, it outputs an update key hsk_t for period t. We write $\mathsf{hsk}_t \leftarrow \Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \mod 2})$.

- $\mathsf{Update}(t, \mathsf{usk}_{t-1}, \mathsf{hsk}_t)$: The *user key-update* algorithm is run by the user at the beginning of each period. Taking as input a period index t, the user secret key usk_{t-1} for period $t-1$ and the update key hsk_t, it returns the user secret key usk_t for period t. We write $\mathsf{usk}_t \leftarrow \mathsf{Update}(t, \mathsf{usk}_{t-1}, \mathsf{hsk}_t)$.
- $\mathsf{Enc}(\mathsf{pk}, t, m)$: The *encryption* algorithm takes as input the public key pk, a period index t and a message $m \in \mathcal{M}$. It outputs a ciphertext $\mathtt{CT}$. We write $\mathtt{CT} \leftarrow \mathsf{Enc}(\mathsf{pk}, t, m)$.
- $\mathsf{Dec}(\mathsf{usk}_t, \mathtt{CT})$: The *decryption* algorithm takes as input a ciphertext $\mathtt{CT}$ under period index t, and the matching user secret key usk_t. It outputs a plaintext m (or "$\perp$" if $\mathtt{CT}$ is invalid). We write $m \leftarrow \mathsf{Dec}(\mathsf{usk}_t, \mathtt{CT})$.

Key-insulated security. This security notion captures the intuition that, if an adversary does not compromise the helper, exposure of the user secret keys for some periods does not affect other periods; furthermore, if a single helper is broken into while a given period t is exposed, only one other period adjacent to t is exposed (recall that even strong key-insulated KIE schemes collapse in this scenario). We refer to this security as $\mathsf{IND\text{-}KI\text{-}ATK}$ where $\mathsf{ATK} \in \{\mathsf{CCA}, \mathsf{CPA}\}$. For an adversary $\mathcal{A}$, we consider the following experiment:

$$\mathbf{Exp}^{\mathsf{IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda): \big[(\mathsf{pk}, \mathsf{usk}_0, (\mathsf{mst}_1, \mathsf{mst}_0)) \leftarrow \mathsf{KeyGen}(\lambda);\ (m_0, m_1, t^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{u}}(\cdot),\mathcal{O}_{\mathrm{h}}(\cdot),\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{find}}(\mathsf{pk});$$
$$\beta \xleftarrow{\$} \{0,1\}; \mathtt{CT}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, t^*, m_\beta);\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{u}}(\cdot),\mathcal{O}_{\mathrm{h}}(\cdot),\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{guess}}(\mathsf{pk}, \mathtt{CT}^*);\ \text{return 1 if } \beta = \beta' \text{ or 0 otherwise}\big],$$

where $\mathcal{O}_{\mathrm{u}}(\cdot)$ is a user secret key oracle which for given a period index t returns the user secret key usk_t, $\mathcal{O}_{\mathrm{h}}(\cdot)$ is a helper key oracle which for given an index $i \in \{0,1\}$ returns the helper key mst_i, and $\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)$ is a decryption oracle which for given $(t, \mathtt{CT})$ returns $m \leftarrow \mathsf{Dec}(\mathsf{usk}_t, \mathtt{CT})$ if $\mathsf{ATK} = \mathsf{CCA}$, or "$\perp$" if $\mathsf{ATK} = \mathsf{CPA}$. It is mandated that $|m_0| = |m_1|$ and the following requirements are satisfied:

(1). $\mathcal{A}$ cannot issue the user secret key query $\mathcal{O}_{\mathrm{u}}(t^*)$;
(2). $\mathcal{A}$ cannot issue both queries $\mathcal{O}_{\mathrm{u}}(t^*-1)$ and $\mathcal{O}_{\mathrm{h}}(t^* \mod 2)$;
(3). $\mathcal{A}$ cannot issue both the helper key queries $\mathcal{O}_{\mathrm{h}}(1)$ and $\mathcal{O}_{\mathrm{h}}(0)$;
(4). if $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{A}$ cannot issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*, \mathtt{CT}^*)$.

We define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathsf{IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda) = 1] - \frac{1}{2}\right|$.

Remark 2. This experiment considers two kinds of adversaries: Type I attackers who do not corrupt helpers during the simulation. In contrast, Type II adversaries corrupt exactly one helper without corrupting a user secret key that would trivially expose the target period t^*.

Remark 3. In the previous security model for PKIE [26, 36], it is additionally required that the adversary cannot issue both queries $\mathcal{O}_{\mathrm{u}}(t^*+1)$ and $\mathcal{O}_{\mathrm{h}}((t^*+1) \mod 2)$. We also notice that, in previous PKIE schemes [26, 36], the adversary is able to derive usk_{t^*} if she corrupts both usk_{t^*+1} and $\mathsf{mst}_{(t^*+1) \mod 2}$. However, according to the definition of PKIE, corruption of usk_{t^*+1} and $\mathsf{mst}_{(t^*+1) \mod 2}$ does not inherently mean the exposure of usk_{t^*}. So, in our security model, we do not impose this restriction on the adversary. As can be seen later, in our PKIE schemes, it is impossible for the adversary to derive usk_{t^*} even if she corrupts both usk_{t^*+1} and $\mathsf{mst}_{(t^*+1) \mod 2}$.

Definition 3 *A* $\mathtt{PKIE}$ *scheme is said to be* $\mathsf{IND\text{-}KI\text{-}CCA}$ *(resp.* $\mathsf{IND\text{-}KI\text{-}CPA}$*) secure, if there exists no PPT adversary* $\mathcal{A}$ *who has non-negligible advantage* $\mathrm{Adv}^{\mathsf{IND\text{-}KI\text{-}CCA}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{IND\text{-}KI\text{-}CPA}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)$*).*

Strong key-insulated security. Key-insulated security can be further enhanced to cover the compromise of both helper keys. To define this security notion, we first define the notion of $\mathsf{strong\text{-}IND\text{-}KI\text{-}ATK}$ security, where $\mathsf{ATK} \in \{\mathsf{CCA}, \mathsf{CPA}\}$. For an adversary $\mathcal{A}$, we consider the following experiment:

$\mathbf{Exp}^{\mathsf{strong\text{-}IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)$:$\big[(\mathsf{pk},\mathsf{usk}_0,(\mathsf{mst}_1,\mathsf{mst}_0)) \leftarrow \mathsf{KeyGen}(\lambda)$; $(m_0,m_1,t^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{find}}(\mathsf{pk},\mathsf{mst}_1,\mathsf{mst}_0)$; $\beta \xleftarrow{\$} \{0,1\}$;$\mathtt{CT}^* \leftarrow \mathsf{Enc}(\mathsf{pk},t^*,m_\beta)$; $\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{guess}}(\mathsf{pk},\mathsf{mst}_1,\mathsf{mst}_0,\mathtt{CT}^*)$; return 1 if $\beta=\beta'$ or 0 otherwise$\big]$,

where $\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)$ is the same as in experiment $\mathbf{Exp}^{\mathsf{IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)$. It is mandated that $|m_0|=|m_1|$, and if $\mathsf{ATK}=\mathsf{CCA}$ then $\mathcal{A}$ cannot issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*,\mathtt{CT}^*)$. We define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathsf{strong\text{-}IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)=\left|\Pr[\mathbf{Exp}^{\mathsf{strong\text{-}IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)=1]-\frac{1}{2}\right|$.

Definition 4 *We say that a* PKIE *scheme is* strong-IND-KI-CCA *(resp.* strong-IND-KI-CPA*) secure, if there is no PPT adversary $\mathcal{A}$ who has non-negligible advantage* $\mathrm{Adv}^{\mathsf{strong\text{-}IND\text{-}KI\text{-}CCA}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{strong\text{-}IND\text{-}KI\text{-}CPA}}_{\mathcal{A},\mathtt{PKIE}}(\lambda)$*).*

Definition 5 *We say that a* PKIE *scheme is strongly key-insulated secure under chosen-ciphertext attack (resp. chosen-plaintext attack), if it is both* IND-KI-CCA *secure (resp.* IND-KI-CPA *secure) and* strong-IND-KI-CCA *secure (resp.* strong-IND-KI-CPA *secure).*

2.5 Bilinear Pairings and Related Complexity Assumptions

In this subsection, we review the definition of bilinear pairings and some related complexity assumptions on which the security of our concrete PKIE schemes are based.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups with the same prime order p. A bilinear pairing is a map $e:\mathbb{G}\times\mathbb{G}\rightarrow\mathbb{G}_T$ with the following properties:

- Bilinearity: $\forall g_1,g_2\in\mathbb{G},\forall a,b\in\mathbb{Z}_p^*$, we have $e(g_1^a,g_2^b)=e(g_1,g_2)^{ab}$;
- Non-degeneracy: There exist $g_1,g_2\in\mathbb{G}$ such that $e(g_1,g_2)\neq 1$;
- Computability: There exists an efficient algorithm to compute $e(g_1,g_2)$ for $\forall g_1,g_2\in\mathbb{G}$.

Definition 6 *The decisional bilinear Diffie-Hellman (DBDH) problem in groups $(\mathbb{G},\mathbb{G}_T)$ is, given a tuple $(g,g^a,g^b,g^c,Z)\in\mathbb{G}^4\times\mathbb{G}_T$ with unknown $a,b,c\xleftarrow{\$}\mathbb{Z}_p^*$, to decide whether $Z=e(g,g)^{abc}$. A polynomial time algorithm $\mathcal{B}$ has advantage ϵ in solving the DBDH problem in groups $(\mathbb{G},\mathbb{G}_T)$, if*

$$\Big|Pr\big[\mathcal{B}(g,g^a,g^b,g^c,Z=e(g,g)^{abc})=1\big]-Pr\big[\mathcal{B}(g,g^a,g^b,g^c,Z=e(g,g)^{d})=1\big]\Big|\geq\epsilon,$$

where the probability is taken over the random choices of a,b,c,d in $\mathbb{Z}_p^*$, the random choice of g in $\mathbb{G}$, and the random bits consumed by $\mathcal{B}$.

We say that the DBDH assumption holds in groups $(\mathbb{G},\mathbb{G}_T)$, if there exists no polynomial time algorithm $\mathcal{B}$ that has non-negligible advantage in solving the DBDH problem in $(\mathbb{G},\mathbb{G}_T)$.

Definition 7 *The ℓ decisional bilinear Diffie-Hellman exponentiation (ℓ-DBDHE) assumption in groups $(\mathbb{G},\mathbb{G}_T)$ is, given a tuple $(g,h,g^a,\cdots,g^{a^\ell},g^{a^{\ell+2}},\cdots,g^{a^{2\ell}},Z)\in\mathbb{G}^{2\ell+1}\times\mathbb{G}_T$ with unknown $a\xleftarrow{\$}\mathbb{Z}_p^*$, to decide whether $Z=e(g,h)^{a^{\ell+1}}$. A polynomial time algorithm $\mathcal{B}$ has advantage ϵ in solving the ℓ-DBDHE problem in groups $(\mathbb{G},\mathbb{G}_T)$, if*

$$\begin{aligned}&\Big|Pr\big[\mathcal{B}(g,h,g^a,\cdots,g^{a^\ell},g^{a^{\ell+2}},\cdots,g^{a^{2\ell}},Z=e(g,h)^{a^{\ell+1}})=1\big]\\&-Pr\big[\mathcal{B}(g,h,g^a,\cdots,g^{a^\ell},g^{a^{\ell+2}},\cdots,g^{a^{2\ell}},Z=e(g,g)^{b})=1\big]\Big|\geq\epsilon,\end{aligned}$$

where the probability is taken over the random choices of a,b in $\mathbb{Z}_p^*$, the random choices of g and h in $\mathbb{G}$, and the random bits consumed by $\mathcal{B}$.

We say that the ℓ-DBDHE assumption holds in groups $(\mathbb{G},\mathbb{G}_T)$, if there exists no polynomial algorithm $\mathcal{B}$ that has non-negligible advantage in solving the ℓ-DBDHE problem in $(\mathbb{G},\mathbb{G}_T)$.

3 Generic Construction of Parallel Key-Insulated Encryption

In this section, we first explain the difficulties in generic constructions of PKIE. Then we introduce a new primitive named one-time forward secure public key encryption (OTFS-PKE). Based on this new primitive, we present a generic construction of PKIE.

3.1 Difficulties in Generic Constructions of PKIE

In [21], Dodis et al. showed that an IBE scheme can be converted to a KIE scheme, by viewing the period index as an "identity" and having the PKG as the helper. However, Hanaoka et al. [26] pointed out that it is non-trivial to construct a PKIE scheme from IBE systems.

To illustrate the difficulties, two important facts should first be kept in mind: On the one hand, according to the definition of PKIE, *only one* (not both) of the helper keys are used to update the user secret key in each period. On the other hand, the user secret key should *simultaneously* contain the update keys generated by mst_0 and mst_1; otherwise, the compromise of a single helper and some periods will harm other periods.

Next, let's review a unsuccessful solution which was previously discussed by Libert et al. [36]. To construct a PKIE scheme by combining two IBE schemes, this solution uses the two PKGs to alternatively act as the helpers for even and odd periods, taking the period indices as identities. For example, a user secret key for an *even* period index t consists of $\mathsf{usk}_t = (sk_{t-1}, sk_t)$, where sk_{t-1} is the secret key generated by the first PKG in the previous period for identity "$t-1$", and sk_t is the secret key generated by the second PKG in current period for identity "t". At first glance, such a solution appears to be feasible. Unfortunately, this is not necessary true, since the user secret key for period t will be exposed by corrupting periods $t-1$ and $t+1$. More specifically, $\mathsf{usk}_t = (sk_{t-1}, sk_t)$ can be derived by *combining* sk_{t-1}, picked from $\mathsf{usk}_{t-1} = (sk_{t-2}, sk_{t-1})$, and sk_t, picked from $\mathsf{usk}_{t+1} = (sk_t, sk_{t+1})$.

The insecurity of the above solution lies in the fact that, the two components generated by distinct PKGs can be individually extracted from the user secret keys, which enables the adversary to assemble another user secret key. Based on this observation, Libert et al. [36] pointed out an intuitive relation between secure PKIE and a category of IBE systems whose key extraction algorithm can be viewed as a signature supporting *aggregation*. Now, the user secret key of the resulting PKIE scheme is the aggregation of two components generated by distinct helpers, so that the individual component cannot be extracted. In fact, both HHI scheme [26] and LQY scheme [36] follow this intuition. However, both HHI scheme and LQY scheme are only concrete, and the aggregation property is not generally satisfied in *all* IBE systems, e.g. [6, 9, 17, 23, 42]. Furthermore, both HHI scheme and LQY scheme cannot satisfy our strengthened security notion, since the adversary is able to derive the user secret key usk_t if she corrupts both usk_{t+1} and mst_j. For example, in HHI scheme, the user secret key for period t is defined to be $\mathsf{usk}_t = H(t-1)^{\mathsf{mst}_j} H(t)^{\mathsf{mst}_i}$, where $j = (t-1) \mod 2$ and $i = t \mod 2$. If the adversary corrupts $\mathsf{usk}_{t+1} = H(t)^{\mathsf{mst}_i} H(t+1)^{\mathsf{mst}_j}$ and mst_j, then she can derive the user secret key for period t by computing $\mathsf{usk}_t = \mathsf{usk}_{t+1} \cdot \frac{H(t-1)^{\mathsf{mst}_j}}{H(t+1)^{\mathsf{mst}_j}} = H(t-1)^{\mathsf{mst}_j} H(t)^{\mathsf{mst}_i}$. Thus the above intuition cannot be utilized to generic construction of PKIE from *any* IBE systems, and we must find a different way.

3.2 A New Primitive: One-Time Forward Secure Public Key Encryption

To present our generic construction of PKIE, we first introduce a new primitive named one-time forward secure public key encryption (OTFS-PKE). Like forward secure public key encryption [14, 32], the lifetime of OTFS-PKE is divided into distinct time periods, and the public key remains fixed throughout the lifetime. However, unlike forward secure public key encryption, the secret key in OTFS-PKE can only be evolved once (this is the reason why we use the terminology "one-time" to name this primitive), and then it needs to be regenerated. For convenience, we shall use a bit $\mathsf{flg} \in \{0, 1\}$ to identify whether a secret key of a given period

can be evolved or not. Concretely, in the beginning of a period t where $t \mod 2 = \mathsf{flg}$, the user regenerates a new secret key d_t (refer to it as an *evolvable* secret key), which can be further evolved in the next period $t+1$. While for a period index t where $t \mod 2 = \overline{\mathsf{flg}}$, the secret key d_t is evolved from the previous secret key, and cannot be further evolved (refer to it as an *evolved* secret key). Formally, an OTFS-PKE scheme consists of the following algorithms:

- $\mathsf{Setup}(\lambda, \mathsf{flg})$: The *setup* algorithm takes as input a security parameter λ and a bit $\mathsf{flg} \in \{0,1\}$. It returns a public key PK and a master key MK. We write $(PK, MK) \leftarrow \mathsf{Setup}(\lambda, \mathsf{flg})$. (Without loss of generality, we assume flg is included in PK.)
- $\mathsf{KeyGen}(PK, MK, t)$: The *key generation* algorithm takes as input PK, MK and a period index t where $t \mod 2 = \mathsf{flg}$. It outputs an evolvable secret key d_t for period t. We write $d_t \leftarrow \mathsf{KeyGen}(PK, MK, t)$.
- $\mathsf{Upd}(PK, t, d_{t-1})$: The *key update* algorithm takes as input PK, a period index t where $t \mod 2 = \overline{\mathsf{flg}}$, and the evolvable secret key d_{t-1} of the previous period. It returns the evolved secret key d_t for period t. We write $d_t \leftarrow \mathsf{Upd}(PK, t, d_{t-1})$.
- $\mathsf{Enc}(PK, t, M)$: The *encryption* algorithm takes as input PK, a period index t and a message $M \in \mathcal{M}$. It returns a ciphertext C (or "$\perp$" if C is invalid). We write $C \leftarrow \mathsf{Enc}(PK, t, M)$.
- $\mathsf{Dec}(d_t, C)$: The *decryption* algorithm takes as input the secret key d_t and a ciphertext C. It returns a message M (or "$\perp$" if C is invalid). We write $M \leftarrow \mathsf{Dec}(d_t, C)$.

The correctness of OTFS-PKE means that, for any $M \in \mathcal{M}$ and any periods t_1 (where $t_1 \mod 2 = \mathsf{flg}$) and t_2 (where $t_2 \mod 2 = \overline{\mathsf{flg}}$), it holds that

$$\mathsf{Dec}(\mathsf{KeyGen}(PK, MK, t_1), \mathsf{Enc}(PK, t_1, M)) = M,$$
$$\mathsf{Dec}(\mathsf{Upd}(PK, t_2, \mathsf{KeyGen}(PK, MK, t_2 - 1)), \mathsf{Enc}(PK, t_2, M)) = M.$$

Roughly speaking, the security requirements for an OTFS-PKE scheme should capture the following intuitions: For a period t^* where $t^* \mod 2 = \overline{\mathsf{flg}}$, exposure of the secret key d_{t^*} does not affect any other period (in particular, it is impossible for an adversary to derive the previous secret key d_{t^*-1}); while exposure of the secret key of period t^* where $t^* \mod 2 = \mathsf{flg}$ should merely affect periods t^* and t^*+1. Next, we begin to define the formal semantic security for OTFS-PKE, and we refer to it as $\mathsf{IND\text{-}FS\text{-}ATK}$ where $\mathsf{ATK} = \{\mathsf{CCA}, \mathsf{CPA}\}$. For an adversary $\mathcal{B}$, we consider the following experiment:

$$\mathbf{Exp}^{\mathsf{IND\text{-}FS\text{-}ATK}}_{\mathcal{B},\mathtt{OTFS\text{-}PKE}}(\lambda): \big[\mathsf{flg} \leftarrow \mathcal{B}(\lambda); (PK, MK) \leftarrow \mathsf{Setup}(\lambda, \mathsf{flg});\ (M_0, M_1, t^*) \leftarrow \mathcal{B}^{\mathcal{O}_{\mathrm{ke}}(\cdot), \mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{find}}(PK);$$
$$\theta \xleftarrow{\$} \{0,1\}; C^* \leftarrow \mathsf{Enc}(PK, t^*, M_\theta);\ \theta' \leftarrow \mathcal{B}^{\mathcal{O}_{\mathrm{ke}}(\cdot), \mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{guess}}(PK, C^*);\ \text{return 1 if } \theta = \theta' \text{ or 0 otherwise}\big],$$

where $\mathcal{O}_{\mathrm{ke}}(\cdot)$ is a key-exposure oracle which on input index t returns $d_t \leftarrow \mathsf{KeyGen}(PK, MK, t)$ if $t \mod 2 = \mathsf{flg}$ or $d_t \leftarrow \mathsf{Upd}(PK, t, \mathsf{KeyGen}(PK, MK, t-1))$ if $t \mod 2 = \overline{\mathsf{flg}}$, and $\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)$ is a decryption oracle which on input (t, C) returns $m \leftarrow \mathsf{Dec}(d_t, C)$ if $\mathsf{ATK} = \mathsf{CCA}$, or "$\perp$" if $\mathsf{ATK} = \mathsf{CPA}$. It is mandated that $|M_0| = |M_1|$, and the following requirements should be satisfied:

(1). $\mathcal{B}$ cannot issue the key-exposure query $\mathcal{O}_{\mathrm{ke}}(t^*)$;
(2). If $t^* \mod 2 = \overline{\mathsf{flg}}$, $\mathcal{B}$ cannot issue the key-exposure query $\mathcal{O}_{\mathrm{ke}}(t^*-1)$;
(3). If $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{B}$ cannot issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*, C^*)$.

We define $\mathcal{B}$'s advantage as $\mathrm{Adv}^{\mathsf{IND\text{-}FS\text{-}ATK}}_{\mathcal{B},\mathtt{OTFS\text{-}PKE}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}FS\text{-}ATK}}_{\mathcal{B},\mathtt{OTFS\text{-}PKE}}(\lambda) = 1] - \frac{1}{2}\right|$.

Definition 8 *We say that an* $\mathtt{OTFS\text{-}PKE}$ *scheme is* $\mathsf{IND\text{-}FS\text{-}CCA}$ *(resp.* $\mathsf{IND\text{-}FS\text{-}CPA}$*) secure, if there exists no PPT adversary* $\mathcal{B}$ *who has non-negligible advantage* $\mathrm{Adv}^{\mathsf{IND\text{-}FS\text{-}CCA}}_{\mathcal{B},\mathtt{OTFS\text{-}PKE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{IND\text{-}FS\text{-}CPA}}_{\mathcal{B},\mathtt{OTFS\text{-}PKE}}(\lambda)$*).*

3.3 Generic Construction of PKIE from OTFS-PKE

In this section, we first show how to generically construct a chosen-plaintext secure PKIE scheme from OTFS-PKE and PKE. Then we discuss how to achieve the chosen-ciphertext security.

3.3.1 CPA-Secure Construction

Basic Idea. We first explain how to use two OTFS-PKE schemes to construct a PKIE scheme with (non-strong) key-insulated security. The basic idea of our construction is as follows. We use two OTFS-PKE schemes: $\texttt{OTFS-PKE}_1$ with $\mathsf{flg} = 1$ and $\texttt{OTFS-PKE}_0$ with $\mathsf{flg} = 0$. The master key MK_1 (resp. MK_0) in $\texttt{OTFS-PKE}_1$ (resp. $\texttt{OTFS-PKE}_0$) acts as the helper key mst_1 (resp. mst_0) for the resulting PKIE scheme. In a given period t (let $i = t \mod 2$ and $j = (t-1) \mod 2$), the user secret key is of the form $\mathsf{usk}_t = (d_{j,t}, d_{i,t})$, where $d_{i,t}$ is an *evolvable* secret key directly generated by mst_i, and $d_{j,t}$ is an *evolved* secret key evolved from $d_{j,t-1}$, which is directly generated by mst_j in the previous period. In the next period $t+1$, the helper key mst_j generates a new *evolvable* secret key $d_{j,t+1}$, while $d_{i,t}$ evolves into an *evolved* secret key $d_{i,t+1}$. And hence the user secret key for period $t+1$ is $\mathsf{usk}_{t+1} = (d_{i,t+1}, d_{j,t+1})$. Due to the "forward security" of the OTFS-PKE scheme, it is impossible to derive $d_{i,t}$ from $d_{i,t+1}$, thus it implies that: (1) unlike the unsuccessful solution mentioned in Section 3.1, even if both $\mathsf{usk}_{t-1} = (d_{i,t-1}, d_{j,t-1})$ and $\mathsf{usk}_{t+1} = (d_{i,t+1}, d_{j,t+1})$ are corrupted, $\mathsf{usk}_t = (d_{j,t}, d_{i,t})$ are still unexposed; (2) unlike HHI scheme and LQY scheme, even if the adversary corrupts $\mathsf{usk}_{t+1} = (d_{i,t+1}, d_{j,t+1})$ and mst_j, it is still impossible to derive $\mathsf{usk}_t = (d_{j,t}, d_{i,t})$.

The above resulting PKIE scheme cannot achieve the strong key-insulated security, since the corruption of both helper keys means all the periods will be exposed. To achieve the strong key-insulated security, we use an additional PKE scheme. Suppose the secret key of the PKE scheme is sk, then the user secret key for period t is of the form $\mathsf{usk}_t = (sk, d_{j,t}, d_{i,t})$. Now, even if both of the helper keys are corrupted, the security of all the periods are still ensured, since sk is unknown to the adversary. The detailed construction is shown in Figure 1.

<table>
<tr><td colspan="2">$\mathsf{KeyGen}(\lambda)$: Given a security parameter λ,
1. choose a PKE scheme $\texttt{PKE}$ and two OTFS-PKE schemes $\texttt{OTFS-PKE}_1$ and $\texttt{OTFS-PKE}_0$,
2. run $(pk, sk) \leftarrow \texttt{PKE}.\mathsf{KGen}(\lambda)$,$(PK_1, MK_1) \leftarrow \texttt{OTFS-PKE}_1.\mathsf{Setup}(\lambda, 1)$ and $(PK_0, MK_0) \leftarrow \texttt{OTFS-PKE}_0.\mathsf{Setup}(\lambda, 0)$,
3. run $d_{1,-1} \leftarrow \texttt{OTFS-PKE}_1.\mathsf{KeyGen}(PK_1, MK_1, -1)$, $d_{1,0} \leftarrow \texttt{OTFS-PKE}_1.\mathsf{Upd}(PK_1, 0, d_{1,-1})$ and $d_{0,0} \leftarrow \texttt{OTFS-PKE}_0.\mathsf{KeyGen}(PK_0, MK_0, 0)$,
4. output $\mathsf{pk} = (pk, PK_1, PK_0)$, $\mathsf{usk}_0 = (sk, d_{1,0}, d_{0,0})$, $\mathsf{mst}_1 = MK_1$ and $\mathsf{mst}_0 = MK_0$.</td></tr>
<tr><td>$\Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \mod 2})$: To generate the update key hsk_t with the matching helper key $\mathsf{mst}_{t \mod 2}$,
1. let $i = t \mod 2$,
2. run $d_{i,t} \leftarrow \texttt{OTFS-PKE}_i.\mathsf{KeyGen}(PK_i, \mathsf{mst}_i, t)$,
3. output the update key for period t as $\mathsf{hsk}_t = d_{i,t}$.</td><td>$\mathsf{Update}(t, \mathsf{usk}_{t-1}, \mathsf{hsk}_t)$: In period t, to update the user secret key from usk_{t-1} to usk_t,
1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. parse usk_{t-1} as $(sk, d_{i,t-1}, d_{j,t-1})$ and hsk_t as $d_{i,t}$,
3. run $d_{j,t} \leftarrow \texttt{OTFS-PKE}_j.\mathsf{Upd}(PK_j, t, d_{j,t-1})$,
4. return $\mathsf{usk}_t = (sk, d_{j,t}, d_{i,t})$.</td></tr>
<tr><td>$\mathsf{Enc}(\mathsf{pk}, t, m)$: In period t, to encrypt a message m under public key pk,
1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. pick $M', M'_j \xleftarrow{\$} \mathcal{M}$, and set $M'_i = m \oplus M'_j \oplus M'$,
3. run $C \leftarrow \texttt{PKE}.\mathsf{Enc}(pk, M')$,
$C_j \leftarrow \texttt{OTFS-PKE}_j.\mathsf{Enc}(PK_j, t, M'_j)$,
$C_i \leftarrow \texttt{OTFS-PKE}_i.\mathsf{Enc}(PK_i, t, M'_i)$,
4. return $\texttt{CT} = (C, C_j, C_i)$.</td><td>$\mathsf{Dec}(\texttt{CT}, \mathsf{usk}_t)$: To decrypt a ciphertext $\texttt{CT}$ with the matching user secret key usk_t,
1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. parse $\texttt{CT}$ as (C, C_j, C_i) and usk_t as $(sk, d_{j,t}, d_{i,t})$,
3. run $M' \leftarrow \texttt{PKE}.\mathsf{Dec}(sk, C)$,
$M'_j \leftarrow \texttt{OTFS-PKE}_j.\mathsf{Dec}(d_{j,t}, C_j)$,
$M'_i \leftarrow \texttt{OTFS-PKE}_i.\mathsf{Dec}(d_{i,t}, C_i)$,
4. return $m = M' \oplus M'_j \oplus M'_i$.</td></tr>
</table>

Figure 1: Generic Construction of PKIE from OTFS-PKE and PKE

Remark 4. In the above construction, we assume that the PKE scheme and the OTFS-PKE schemes have the same message space. Otherwise, we can use proper encoding functions or the

standard KEM/DEM framework to satisfy this requirement.

Theorem 1 *The above* PKIE *scheme is strongly key-insulated secure under chosen-plaintext attack, if the underlying* PKE *scheme is* IND-CPA*-secure and the underlying* OTFS-PKE *schemes are* IND-FS-CPA*-secure.*

Theorem 1 follows directly from the following Lemmas 1 and 2.

Lemma 1 *The above* PKIE *scheme is* IND-KI-CPA *secure, if the* OTFS-PKE *schemes* $\texttt{OTFS-PKE}_1$ *and* $\texttt{OTFS-PKE}_0$ *are* IND-FS-CPA *secure.*

Proof. Given a t-time adversary $\mathcal{A}$ who can break the IND-KI-CPA security of our generic PKIE scheme with advantage ϵ, we indicate that we can construct a $(t + q_u\tau_{\mathrm{ku}})$-time adversary $\mathcal{B}$ that can break the IND-FS-CPA-security of the OTFS-PKE schemes with advantage $\epsilon/8$, where τ_{ku} denotes the running time of algorithms KeyGen and Upd in the OTFS-PKE schemes, and q_u denote the number of oracle $\mathcal{O}_u$ queries. Here $\mathcal{B}$ act as two principals: on the one hand, he is the *challenger* for adversary $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\text{IND-KI-CPA}}_{\mathcal{A},\texttt{PKIE}}$; on the other hand, he is the *adversary* in the experiment $\mathbf{Exp}^{\text{IND-FS-CPA}}_{\mathcal{B},\texttt{OTFS-PKE}}$.

As explained in Remark 2, two kinds of adversaries should be distinguished. So, before starting the simulation, $\mathcal{B}$ first tosses a coin $\mathcal{COIN} \xleftarrow{\$} \{0,1\}$ to guess which kind of adversary $\mathcal{A}$ will be. If $\mathcal{COIN} = 0$, it expects to face a Type I adversary. If $\mathcal{COIN} = 1$, it forecasts a Type II adversary.

If $\mathcal{COIN} = 0$, below shows how $\mathcal{B}$ acts as the challenger and provides the simulations for $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\text{IND-KI-CPA}}_{\mathcal{A},\texttt{PKIE}}$, with the help of his challenger in the experiment $\mathbf{Exp}^{\text{IND-FS-CPA}}_{\mathcal{B},\texttt{OTFS-PKE}}$:

`Init Stage.` $\mathcal{B}$ first picks a random bit $c \xleftarrow{\$} \{0,1\}$ to guess $t^* \mod 2 = c$, where t^* is the target period index output by $\mathcal{A}$ later. In the following, $\mathcal{B}$ will challenge with the OTFS-PKE scheme $\texttt{OTFS-PKE}_c$ with $\mathsf{flg} = c$, and then $\mathcal{B}$ is given the public key PK_c. $\mathcal{B}$ chooses a PKE scheme PKE and another OTFS-PKE scheme $\texttt{OTFS-PKE}_{\overline{c}}$, where $\overline{c}$ denotes $1 - c$. Then $\mathcal{B}$ runs $(PK_{\overline{c}}, MK_{\overline{c}}) \leftarrow \texttt{OTFS-PKE}_{\overline{c}}.\mathsf{Setup}(\lambda, \overline{c})$ and $(pk, sk) \leftarrow \texttt{PKE}.\mathsf{KeyGen}(\lambda)$. Finally, $\mathcal{B}$ gives $pk, PK_{\overline{c}}$ and PK_c as the public key pk to $\mathcal{A}$, and keeps sk and $MK_{\overline{c}}$ to himself.

`Find Stage.` In this stage, $\mathcal{A}$ issues a series of queries, and $\mathcal{B}$ responds as follows:

- User secret key oracle $\mathcal{O}_{\mathrm{u}}(t)$: $\mathcal{B}$ responds to this query as follows:
  1. Issue a key-exposure query $\mathcal{O}_{\mathrm{ke}}(t)$ to his challenger, and then is given the secret key $d_{c,t}$.
  2. If $t \mod 2 = \overline{c}$, run $d_{\overline{c},t} \leftarrow \texttt{OTFS-PKE}_{\overline{c}}.\mathsf{KeyGen}(PK_{\overline{c}}, MK_{\overline{c}}, t)$, and return $\mathsf{usk}_t = (sk, d_{c,t}, d_{\overline{c},t})$ to $\mathcal{A}$.
  3. Otherwise, first run $d_{\overline{c},t-1} \leftarrow \texttt{OTFS-PKE}_{\overline{c}}.\mathsf{KeyGen}(PK_{\overline{c}}, MK_{\overline{c}}, t-1)$, and then run $d_{\overline{c},t} \leftarrow \texttt{OTFS-PKE}_{\overline{c}}.\mathsf{Upd}(PK_{\overline{c}}, t, d_{\overline{c},t-1})$. Return $\mathsf{usk}_t = (sk, d_{\overline{c},t}, d_{c,t})$ to $\mathcal{A}$.
- Helper key query $\mathcal{O}_{\mathrm{h}}(i)$ with $i \in \{0,1\}$: $\mathcal{B}$ outputs a random bit and **aborts**, since it means that $\mathcal{B}$ guessed the wrong $\mathcal{COIN}$.

`Challenge.` Once $\mathcal{A}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. If $t^* \mod 2 = \overline{c}$, $\mathcal{B}$ outputs a random bit and **aborts**, since it means that $\mathcal{B}$ guessed the wrong c. Otherwise, $\mathcal{B}$ performs the following steps:

1. Randomly choose $M', M'_{\overline{c}} \in \mathcal{M}$, and set $M_{c,0} = m_0 \oplus M' \oplus M'_{\overline{c}}$ and $M_{c,1} = m_1 \oplus M' \oplus M'_{\overline{c}}$.
2. Submit $(M_{c,0}, M_{c,1})$ and t^* to his challenger, and then is given the corresponding challenge ciphertext C^*_c.

3. Compute $C_{\overline{c}} \leftarrow \texttt{OTFS-PKE}_{\overline{c}}.\mathsf{Enc}(PK_{\overline{c}}, t^*, M'_{\overline{c}})$ and $C \leftarrow \texttt{PKE}.\mathsf{Enc}(pk, M')$.
4. Finally, return $\mathrm{CT}^* = (C, C_{\overline{c}}, C_c^*)$ to $\mathcal{A}$.

`Guess stage.` $\mathcal{A}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{B}$ also responds as in find stage.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. Then $\mathcal{B}$ simply outputs $\theta' = \beta'$ as his guess for his experiment $\mathbf{Exp}^{\mathsf{IND\text{-}FS\text{-}CPA}}_{\mathcal{B},\texttt{OTFS-PKE}}$.

If $\mathcal{COIN} = 1$, $\mathcal{B}$ provides the simulation for $\mathcal{A}$ as follows:

`Init Stage.` $\mathcal{B}$ first picks a random bit $b \xleftarrow{\$} \{0,1\}$ to guess that $\mathcal{A}$ will corrupt the helper key $\mathsf{mst}_{\overline{b}}$, where $\overline{b}$ denotes $1-b$. In the following, $\mathcal{B}$ will challenge with the OTFS-PKE scheme $\texttt{OTFS-PKE}_b$ with $\mathsf{flg} = b$, and then $\mathcal{B}$ is given the public key PK_b. $\mathcal{B}$ chooses another OTFS-PKE system $\texttt{OTFS-PKE}_{\overline{b}}$ as well as a PKE system $\texttt{PKE}$, and runs $(PK_{\overline{b}}, MK_{\overline{b}}) \leftarrow \texttt{OTFS-PKE}_{\overline{b}}.\mathsf{Setup}(\lambda, \overline{b})$ and $(pk, sk) \leftarrow \texttt{PKE}.\mathsf{KeyGen}(\lambda)$. Then, $\mathcal{B}$ gives $pk, PK_{\overline{b}}$ and PK_b as the public key pk to $\mathcal{A}$, and keeps sk and $MK_{\overline{b}}$ to himself.

`Find Stage.` In this stage, $\mathcal{A}$ issues a series of queries, and $\mathcal{B}$ responds as follows:

- User secret key oracle $\mathcal{O}_{\mathrm{u}}(t)$: $\mathcal{B}$ responds to this query for $\mathcal{A}$ as follows:
  1. Issue a key-exposure query $\mathcal{O}_{\mathrm{ke}}(t)$ to his challenger, and then is given the secret key $d_{b,t}$.
  2. If $t \mod 2 = \overline{b}$, run $d_{\overline{b},t} \leftarrow \texttt{OTFS-PKE}_{\overline{b}}.\mathsf{KeyGen}(PK_{\overline{b}}, MK_{\overline{b}}, t)$, and return $\mathsf{usk}_t = (sk, d_{b,t}, d_{\overline{b},t})$ to $\mathcal{A}$.
  3. Otherwise, first run $d_{\overline{b},t-1} \leftarrow \texttt{OTFS-PKE}_{\overline{b}}.\mathsf{KeyGen}(PK_{\overline{b}}, MK_{\overline{b}}, t-1)$, and then $d_{\overline{b},t} \leftarrow \texttt{OTFS-PKE}_{\overline{b}}.\mathsf{Upd}(PK_{\overline{b}}, t, d_{\overline{b},t-1})$. Return $\mathsf{usk}_t = (sk, d_{\overline{b},t}, d_{b,t})$ to $\mathcal{A}$.
- Helper key query $\mathcal{O}_{\mathrm{h}}(i)$ with $i \in \{0,1\}$: If $i = b$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ returns $\mathsf{mst}_i = MK_{\overline{b}}$ to $\mathcal{A}$.

`Challenge.` Once $\mathcal{A}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathcal{M}$ on which it wishes to be challenged. Then $\mathcal{B}$ performs the following steps:

1. Randomly choose $M', M'_{\overline{b}} \in \mathcal{M}$, and set $M_{b,0} = m_0 \oplus M' \oplus M'_{\overline{b}}$ and $M_{b,1} = m_1 \oplus M' \oplus M'_{\overline{b}}$.
2. Submit $(M_{b,0}, M_{b,1})$ and t^* to his challenger, and then is given the corresponding challenge ciphertext C_b^*.
3. Compute $C_{\overline{b}} \leftarrow \texttt{OTFS-PKE}_{\overline{b}}.\mathsf{Enc}(PK_{\overline{b}}, t^*, M'_{\overline{b}})$ and $C \leftarrow \texttt{PKE}.\mathsf{Enc}(pk, M')$.
4. If $t^* \mod 2 = b$ then return $\mathrm{CT}^* = (C, C_{\overline{b}}, C_b^*)$ to $\mathcal{A}$; otherwise return $\mathrm{CT}^* = (C, C_b^*, C_{\overline{b}})$ to $\mathcal{A}$.

`Guess stage.` $\mathcal{A}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{B}$ also responds as in find stage.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. Then $\mathcal{B}$ simply outputs $\theta' = \beta'$ as his guess for his experiment $\mathbf{Exp}^{\mathsf{IND\text{-}FS\text{-}CPA}}_{\mathcal{B},\texttt{OTFS-PKE}}$.

Analysis: The simulations provided for $\mathcal{A}$ are perfect, unless the following events happens:

E_1: a helper key query $\mathcal{O}_{\mathrm{h}}(i)$ was issued when $\mathcal{COIN} = 0$
E_2: the helper key query $\mathcal{O}_{\mathrm{h}}(b)$ was issued when $\mathcal{COIN} = 1$
E_3: the user secret key query $\mathcal{O}_{\mathrm{u}}(t^*-1)$ was issued when $t^* \mod 2 = \overline{c}$ and $\mathcal{COIN} = 0$

To analyze the above events, we alternately consider the following events:

H_1: $\mathcal{B}$ successfully guesses the kind of attack produced by $\mathcal{A}$
H_2: $\mathcal{B}$ successfully guesses the value of $t^* \mod 2$ when $\mathcal{COIN} = 0$
H_3: $\mathcal{B}$ luckily predicts which helper's key is exposed when $\mathcal{COIN} = 1$

Clearly, we have $\Pr[H_1] = \Pr[H_2] = \Pr[H_3] = 1/2$. Also, we have $H_1 \wedge H_2 \wedge H_3 \Rightarrow \neg E_1 \wedge \neg E_2 \wedge \neg E_3$. The conjunction of events H_1, H_2 and H_3 is readily seen to occur with probability greater than $1/8$, and hence $\mathcal{B}$'s advantage is greater than $\epsilon/8$. It is also clear that $\mathcal{B}$'s running time is bounded by $t + q_u\tau_{\mathrm{ku}}$. Thus the proof of Lemma 1 is concluded. □

Lemma 2 *The above* PKIE *scheme is* strong-IND-KI-CPA *secure, if the* PKE *scheme* PKE *is* IND-CPA *secure.*

Proof. Suppose that there exists a t-time adversary $\mathcal{A}$ that breaks the strong-IND-KI-CPA security of our generic PKIE scheme with advantage ϵ, then we can make use of $\mathcal{A}$ to construct a t-time adversary $\mathcal{B}$ that can break the IND-CPA-security of the PKE scheme PKE with the same advantage. Given the public key pk of the PKE scheme PKE, $\mathcal{B}$ acts as the challenger and provides the simulations for $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\text{strong-IND-KI-CPA}}_{\mathcal{A},\text{PKIE}}$ as below:

`Init Stage.` $\mathcal{B}$ first chooses two OTFS-PKE schemes OTFS-PKE_1 and OTFS-PKE_0, and runs $(PK_1, MK_1) \leftarrow \text{OTFS-PKE}_1.\mathsf{Setup}(\lambda, 1)$ and $(PK_0, MK_0) \leftarrow \text{OTFS-PKE}_0.\mathsf{Setup}(\lambda, 0)$. Then, $\mathcal{B}$ gives the public key $\mathsf{pk} = (pk, PK_1, PK_0)$ as well as the two helper keys $\mathsf{mst}_1 = MK_1$ and $\mathsf{mst}_0 = MK_0$ to $\mathcal{A}$.

`Challenge.` $\mathcal{A}$ outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathcal{M}$. $\mathcal{B}$ performs the following steps:

1. $\mathcal{B}$ randomly chooses two equal-length messages $M_0', M_1' \in \mathcal{M}$, and defines $M_0 = m_0 \oplus M_0' \oplus M_1'$ and $M_1 = m_1 \oplus M_0' \oplus M_1'$.
2. $\mathcal{B}$ submits (M_0, M_1) and t^* to his challenger, and then is given the corresponding challenge ciphertext C^*.
3. $\mathcal{B}$ computes $C_1 \leftarrow \text{OTFS-PKE}_1.\mathsf{Enc}(PK_1, t^*, M_1')$ and $C_0 \leftarrow \text{OTFS-PKE}_0.\mathsf{Enc}(PK_0, t^*, M_0')$.
4. If $t^* \mod 2 = 1$, then $\mathcal{B}$ returns $\mathrm{CT}^* = (C^*, C_0, C_1)$ to $\mathcal{A}$; else returns $\mathrm{CT}^* = (C^*, C_1, C_0)$ to $\mathcal{A}$.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0, 1\}$. Then $\mathcal{B}$ simply outputs $\delta' = \beta'$ as his guess for his experiment $\mathbf{Exp}^{\text{IND-FS-CPA}}_{\mathcal{B},\text{OTFS-PKE}}$.

Clearly, the simulations provided for $\mathcal{A}$ are perfect. It is also clear that $\mathcal{B}$'s advantage is ϵ, and his running time is bounded by t. □

3.3.2 Chosen-ciphertext Security

In our generic PKIE scheme, the algorithm Enc is a multiple encryption, and only achieves the chosen-plaintext security. Using Dodis et al.'s technique [20], we can readily achieve the chosen-ciphertext security for our scheme. Since this is straightforward, we here do not provide the detailed construction.

3.4 Generic Constructions of OTFS-PKE

In this section, we first show that OTFS-PKE can be generically constructed from any IBE scheme. Furthermore, we point out that it is also possible to directly construct OTFS-PKE from any 2-HIBE without degenerating into IBE.

3.4.1 Construction of OTFS-PKE from IBE

In this subsection, we demonstrate a generic construction of OTFS-PKE from arbitrary IBE.

Our basic idea is as follows. We set the public parameter $param$ and the master secret key msk in the IBE scheme as the public key PK and the master key MK in the OTFS-PKE scheme, respectively. In period t where $t = \mathsf{flg}$, the evolvable secret key is $d_t = (sk_t, sk_{t+1})$, which consists of two secret keys for identities "t" and "$t+1$". Encryption is only carried under identity "t", and hence the decryption only involves sk_t while sk_{t+1} is merely used for evolution. That is, in the next period, the secret key evolves to be $d_{t+1} = sk_{t+1}$, where sk_t has been deleted. Observe that from $d_{t+1} = sk_{t+1}$, it is impossible to derive the decryption key sk_t in the previous secret key. Therefore, the security of period t is still ensured even if the period $t+1$ is exposed. Recall that this is exactly the requirement for a *secure* OTFS-PKE scheme. Detailed construction is presented in Figure 2.

<table>
<tr><td colspan="2">$\mathsf{Setup}(\lambda, \mathsf{flg})$: Given a security parameter λ and a bit $\mathsf{flg} \in \{0,1\}$,
1. run $(param, msk) \leftarrow \mathtt{IBE.Setup}(\lambda)$, and return $PK = (param, \mathsf{flg})$ and $MK = msk$.</td></tr>
<tr><td>$\mathsf{KeyGen}(PK, MK, t)$: To generate an evolvable secret key for period t (where $t \mod 2 = \mathsf{flg}$),

1. run $sk_t \leftarrow \mathtt{IBE.Extract}(PK, MK, t)$,
$sk_{t+1} \leftarrow \mathtt{IBE.Extract}(PK, MK, t+1)$,
2. return $d_t = (sk_t, sk_{t+1})$.</td><td>$\mathsf{Upd}(PK, t, d_{t-1})$: In period t where $t \mod 2 = \overline{\mathsf{flg}}$, to generate an evolved secret key for period t,

1. parse d_{t-1} as (sk_{t-1}, sk_t),
2. return $d_t = sk_t$.</td></tr>
<tr><td>$\mathsf{Enc}(PK, t, m)$: In period t, to encrypt a message m under public key PK,

1. run $C \leftarrow \mathtt{IBE.Enc}(PK, t, m)$,
2. return the ciphertext C.</td><td>$\mathsf{Dec}(d_t, C)$: To decrypt a ciphertext C using secret key d_t,

1. parse d_t as $d_t = sk_t$ (if $t \mod 2 = \overline{\mathsf{flg}}$)
or $d_t = (sk_t, sk_{t+1})$ (if $t \mod 2 = \mathsf{flg}$),
2. return $m \leftarrow \mathtt{IBE.Dec}(sk_t, C)$.</td></tr>
</table>

Figure 2: Generic Construction of OTFS-PKE from IBE

Theorem 2 *The above* OTFS-PKE *scheme is* IND-FS-ATK *secure, if the underlying* IBE *scheme* `IBE` *is* IND-ID-ATK *secure, where* $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

Proof. Suppose that there exists a t-time adversary $\mathcal{B}$ who can break the IND-FS-ATK-security of our IBE-based generic OTFS-PKE scheme, then we can make use of $\mathcal{B}$ to construct a t-time adversary $\mathcal{A}$ who can break the IND-ID-ATK-security of the `IBE` scheme with the same advantage. $\mathcal{A}$ provides the simulations for $\mathcal{B}$ as follows:

`Init Stage.` $\mathcal{B}$ first commits to a bit $\mathsf{flg} \in \{0,1\}$. Then $\mathcal{A}$ obtains the public parameter $param$ from his challenger, and forwards $PK = (param, \mathsf{flg})$ to $\mathcal{B}$.

`Find Stage.` In this stage, $\mathcal{B}$ issues a series of queries, and $\mathcal{A}$ responds as follows:

- Key-exposure oracle $\mathcal{O}_{\mathrm{ke}}(t)$: To answer this query, two cases should be distinguished:
  - $t = \mathsf{flg}$: $\mathcal{A}$ issues an extraction query $\mathcal{O}_{\mathrm{ext}}(t)$ and $\mathcal{O}_{\mathrm{ext}}(t+1)$ to his challenger, and then is given the secret keys sk_t and sk_{t+1} for identities t and $t+1$ respectively. Then $\mathcal{A}$ forwards $d_t = (sk_t, sk_{t+1})$ to $\mathcal{B}$.
  - $t = \overline{\mathsf{flg}}$: $\mathcal{A}$ issues an extraction query $\mathcal{O}_{\mathrm{ext}}(t)$, and obtains the secret key sk_t. Then $\mathcal{A}$ forwards $d_t = sk_t$ to $\mathcal{B}$.

  If $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{A}$ is additionally allowed to issue the following decryption queries.
- Decryption query $\mathcal{O}_{\mathrm{d}}(t, C)$: Taking t as an identity, $\mathcal{A}$ submits (t, C) to his challenger as a decryption query, and then forwards the result to $\mathcal{B}$.

`Challenge.` Once $\mathcal{B}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $M_0, M_1 \in \mathcal{M}$. $\mathcal{A}$ submits t^* and (M_0, M_1) to his challenger, and then obtains a challenge ciphertext C^*, which is then forwarded to $\mathcal{B}$.

`Guess stage.` $\mathcal{B}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{A}$ also responds as in find stage. Note that, if $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{B}$ is not allowed to issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*, C^*)$.

`Output.` Finally, $\mathcal{B}$ outputs a guess $\theta' \in \{0,1\}$. Then $\mathcal{A}$ simply outputs $\delta' = \theta'$ as his guess for his experiment $\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}ATK}}_{\mathcal{A},\mathtt{IBE}}$.

Clearly, the simulations provided for $\mathcal{B}$ are perfect, and $\mathcal{A}$ has the same advantage as $\mathcal{B}$ within the same time bound. Thus it completes the proof of Theorem 2. □

3.4.2 Construction of OTFS-PKE from 2-HIBE

It is well known that forward secure public key encryption can be obtained by using HIBE [14]. Therefore, our next direction for designing OTFS-PKE is to directly use HIBE as a building block. Since in the previous approach, we can also construct OTFS-PKE from standard IBE (see Sec. 3.4.1), one might think that our HIBE-based construction is not necessary. However, due to the difference between these two approaches, there are concrete OTFS-PKE schemes with interesting properties which cannot be obtained without using our HIBE-based construction (see Sec. 4.2.2). Thus, we consider that HIBE-based construction is worth discussing despite of existence of IBE-based one.

Our basic idea is as follows. For a given 2-HIBE scheme, we have its public parameter $param$ and the master secret key msk as the public key PK and the master key MK for the OTFS-PKE scheme, respectively. In the beginning of a period t (where $t \mod 2 = \mathsf{flg}$), taking the index t as a one-level identity, we use the master secret key msk to generate a secret key sk_t, which is viewed as an *evolvable* secret key for period t. In the next period $t+1$, we use sk_t to generate a secret key $sk_{(t,t+1)}$ for the two-level offspring identity "$(t, t+1)$". Here $sk_{(t,t+1)}$ is viewed as an *evolved* secret key for period $t+1$. Note that according to the property of HIBE, from $sk_{(t,t+1)}$, it is impossible to derive the pervious secret key sk_t. This is exactly the requirement for a *secure* OTFS-PKE scheme. The encryption and decryption algorithms can be accordingly designed. Figure 3 gives our construction:

<table>
<tr><td colspan="2">$\mathsf{Setup}(\lambda, \mathsf{flg})$: Given a security parameter λ and a bit $\mathsf{flg} \in \{0,1\}$,
1. run $(param, msk) \leftarrow \mathtt{HIBE.Setup}(\lambda, 2)$, and return $PK = (param, \mathsf{flg})$ and $MK = msk$.</td></tr>
<tr><td>$\mathsf{KeyGen}(PK, MK, t)$: To generate an evolvable secret key for period t where $t \mod 2 = \mathsf{flg}$,
1. run $sk_t \leftarrow \mathtt{HIBE.Extract}(PK, MK, t)$,
2. return $d_t = sk_t$.</td><td>$\mathsf{Upd}(PK, t, d_{t-1})$: In period t where $t \mod 2 = \overline{\mathsf{flg}}$, to generate an evolved secret key for period t,
1. run $sk_{(t-1,t)} \leftarrow \mathtt{HIBE.Extract}(PK, d_{t-1}, (t-1, t))$,
2. return $d_t = sk_{(t-1,t)}$.</td></tr>
<tr><td>$\mathsf{Enc}(PK, t, m)$: In period t, to encrypt a message m under public key PK,
1. if $t \mod 2 = \mathsf{flg}$, run $C \leftarrow \mathtt{HIBE.Enc}(PK, t, M)$; else run $C \leftarrow \mathtt{HIBE.Enc}(PK, (t-1, t), M)$,
2. return the ciphertext C.</td><td>$\mathsf{Dec}(d_t, C)$: To decrypt a ciphertext C using the secret key d_t,
1. compute $M \leftarrow \mathtt{HIBE.Dec}(d_t, C)$,
2. return M.</td></tr>
</table>

Figure 3: Generic Construction of OTFS-PKE from 2-HIBE

Remark 5. The above 2-HIBE scheme is assumed to have a property that, the intermediate nodes' keys can be used not only for key derivation but also for decryption. Such a property is shared by most of the existing HIBE schemes. We notice that Horwitz and Lynn [30] defined another model for HIBE where the intermediate nodes' keys are only used for key derivation. Note that, by appending a dummy identity (e.g., "$0 \cdots 0$") to a one-level (intermediate) identity, our generic construction can be easily modified to be compatible with this model. Concretely, if one wants to encrypt a message to the one-level identity "t", he encrypts it under the two-level identity "$(t, 0 \cdots 0)$". The decryption can be done by first deriving the secret key $sk_{(t,0\cdots 0)}$ from sk_t, and then using it to decrypt the ciphertext.

Theorem 3 *The above* OTFS-PKE *scheme is* IND-FS-ATK *secure, if the underlying* HIBE *scheme* HIBE *is* IND-ID-ATK *secure, where* $\mathsf{ATK} \in \{\mathsf{CPA}, \mathsf{CCA}\}$.

Proof. Suppose that there exists a t-time adversary $\mathcal{B}$ who can break the IND-FS-ATK-security of our HIBE-based generic OTFS-PKE scheme with advantage ϵ, then we can make use of $\mathcal{B}$ to construct a t-time adversary $\mathcal{A}$ who can break the IND-ID-ATK-security of the HIBE scheme with the same advantage. $\mathcal{A}$ provides the simulations for $\mathcal{B}$ as below:

`Init Stage.` $\mathcal{B}$ first commits to a bit $\mathsf{flg} \in \{0, 1\}$. Then $\mathcal{A}$ obtains the public parameter $param$ from his challenger, and forwards $PK = (param, \mathsf{flg})$ to $\mathcal{B}$.

`Find Stage.` In this stage, $\mathcal{B}$ issues a series of queries, and $\mathcal{A}$ responds as follows:

- Key-exposure oracle $\mathcal{O}_{\mathrm{ke}}(t)$: To answer this query, two cases should be distinguished:
  - $t = \mathsf{flg}$: $\mathcal{A}$ issues an extraction query $\mathcal{O}_{\mathrm{ext}}(t)$ to his challenger, and then is given the secret key sk_t for identity t. $\mathcal{A}$ simply forwards $d_t = sk_t$ to $\mathcal{B}$.
  - $t = \overline{\mathsf{flg}}$: $\mathcal{A}$ issues an extraction query $\mathcal{O}_{\mathrm{ext}}((t-1, t))$, and obtains the secret key $sk_{(t-1,t)}$ for the two-level identity $(t-1, t)$. Then $\mathcal{A}$ returns $d_t = sk_{(t-1,t)}$ to $\mathcal{B}$.

  If $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{A}$ is additionally allowed to issue the following decryption queries.
- Decryption query $\mathcal{O}_{\mathrm{d}}(t, C)$: Taking t as an identity, $\mathcal{A}$ submits (t, C) to his decryption oracle in the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}ATK}}_{\mathcal{A},\mathtt{HIBE}}$, and then forwards the result to $\mathcal{B}$.

`Challenge.` Once $\mathcal{B}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $M_0, M_1 \in \mathcal{M}$. $\mathcal{A}$ prepares the challenge ciphertext for $\mathcal{B}$ according to two cases:

- $t^* = \mathsf{flg}$: $\mathcal{A}$ submits the identity t^* and (M_0, M_1) to his challenger, and obtains a challenge ciphertext C^*, which is then forwarded to $\mathcal{B}$.
- $t^* = \overline{\mathsf{flg}}$: $\mathcal{A}$ submits the two-level identity $(t^* - 1, t^*)$ and (M_0, M_1) to his challenger, and obtains a challenge ciphertext C^*, which is then forwarded to $\mathcal{B}$.

`Guess stage.` $\mathcal{B}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{A}$ also responds as in find stage. Note that, if $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{B}$ is not allowed to issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*, C^*)$.

`Output.` Finally, $\mathcal{B}$ outputs a guess $\theta' \in \{0, 1\}$. Then $\mathcal{A}$ simply outputs $\delta' = \theta'$ as his guess for the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}ID\text{-}ATK}}_{\mathcal{A},\mathtt{HIBE}}$.

Obviously, the simulations provided for $\mathcal{B}$ are perfect. Hence it is clear that $\mathcal{A}$ has the same advantage as $\mathcal{B}$, and his running time is bounded by t. Thus Theorem 3 is proved. □

4 Instantiations of Our Generic Constructions

Previous section indicates that PKIE schemes can be generically constructed from IBE or 2-HIBE. Thus, from existing IBE and HIBE schemes with specific properties, we can construct PKIE schemes with a variety of features which previous PKIE schemes cannot have. In this section, we show such instantiations. There are mainly two kinds of instantiations: (1) constructions from various assumptions, and (2) constructions with better efficiency in certain aspects.

4.1 Instantiations from Various Assumptions

As mentioned above, by using our generic construction, a PKIE scheme can be immediately obtained from arbitrary IBE. This means that if there exists an IBE scheme which is provably secure under some mathematical assumption, then there also exists a PKIE scheme under the same assumption.

For example, based on the lattice-based IBE schemes [16, 24, 37], we can construct PKIE schemes by assuming only difficulty of certain types of lattice problems, e.g. the learning with error problem [41]. These are considered as the first "post-quantum" PKIE schemes.

Furthermore, based on the quadratic-residuosity-based IBE schemes [9,17], we can construct PKIE schemes under the decisional quadratic residuosity assumption. These are considered as the first PKIE schemes based on the factoring problem.

4.2 Efficient Instantiations from Pairings

For example, our IBE-based construction from Boneh-Boyen IBE [6] yields shorter ciphertexts and cheaper cost for encryption and decryption. Furthermore, our HIBE-based construction from Boneh-Boyen-Goh HIBE [7] yields cheaper cost for Δ-Gen algorithm, and this is useful when a helper is a cheap device, e.g. a smart card. It should be also noticed that in terms of computational cost for Δ-Gen, our Boneh-Boyen-Goh-based scheme is more efficient than both the LQY scheme and our Boneh-Boyen-based scheme, and this implies that our HIBE-based generic construction is still useful despite of existence of our IBE-based generic construction.

We remark that, in this subsection, we shall only present the CPA-secure instantiations, since their CCA-security can be achieved by applying the generic techniques in [10, 15] or the direct technique in [11].

In the rest of this paper, we shall use the bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ with prime order $p \geq 2^\lambda$, where λ is the security parameter, and there exists a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$.

4.2.1 Efficient Instantiation from Boneh-Boyen IBE

LQY scheme is based on Boneh-Boyen IBE scheme [6] which is reviewed in Figure 4. Interestingly, when our IBE-based generic construction is instantiated with Boneh-Boyen IBE scheme, we can make use of its algebraic property to obtain a PKIE scheme more efficient than LQY scheme.

$\mathsf{Setup}(\lambda)$: 1. pick $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, $g, g_2, h \xleftarrow{\$} \mathbb{G}$, and set $g_1 = g^\alpha$, 2. output $param = (g, g_1, g_2, h)$ and $msk = g_2^\alpha$.	$\mathsf{Extract}(param, msk, \mathsf{ID})$: 1. pick $r \xleftarrow{\$} \mathbb{Z}_p^*$, 2. output $sk_{\mathsf{ID}} = (a_{\mathsf{ID}}, b_{\mathsf{ID}}) = \left(msk \cdot (g_1^{\mathsf{ID}} h)^r, g^r\right)$.
$\mathsf{Enc}(param, \mathsf{ID}, m)$: 1. pick $s \xleftarrow{\$} \mathbb{Z}_p^*$, 2. compute $C_1 = e(g_1, g_2)^s \cdot m, C_2 = g^s$ and $C_3 = (g_1^{\mathsf{ID}} h)^s$, 3. return $C = (C_1, C_2, C_3)$.	$\mathsf{Dec}(sk_{\mathsf{ID}}, C)$: 1. parse C as (C_1, C_2, C_3) and sk_{ID} as $(a_{\mathsf{ID}}, b_{\mathsf{ID}})$, 2. return $m \leftarrow \dfrac{C_1 \cdot e(C_3, b_{\mathsf{ID}})}{e(C_2, a_{\mathsf{ID}})}$.

Figure 4: Boneh-Boyen IBE scheme [6]

Our concrete PKIE scheme consists of two Boneh-Boyen IBE systems, which share the same the public parameters g, g_2 and h. The master secret keys of the two systems are $\mathsf{mst}_1 = g_2^{\alpha_1}$ and $\mathsf{mst}_0 = g_2^{\alpha_0}$ respectively, which act as the two helper keys for the PKIE scheme and alternately update the user secret keys. It is worth noting that, Boneh-Boyen IBE system has a distinguished *aggregation* property, i.e., a product of secret keys for the same ID under different master keys also works as the secret key for that ID under another master key (which is the product of these underlying master keys). Based on this observation, we product the parameters (i.e., g^{α_1} and g^{α_0}) of the two systems into a single $g_1 = g^{\alpha_1 + \alpha_0}$ (this parameter will be further changed, which will be clear later). Also, two secret keys for the same identity are aggregated into a single one. For example, in the initial user secret key usk_0, the secret key $\left(\mathsf{mst}_1 \cdot (g_1^{H(0)} h)^{r_{1,0}}, g^{r_{1,0}}\right)$, generated by the master secret key mst_1 for identity "0", and the secret key $\left(\mathsf{mst}_0 \cdot (g_1^{H(0)} h)^{r_{0,0}}, g^{r_{0,0}}\right)$, generated by mst_0 for identity "0", are integrated into $(a_{0,0}, b_{0,0}) = \left(\mathsf{mst}_1 \mathsf{mst}_0 \cdot (g_1^{H(0)} h)^{r_{1,0}+r_{0,0}}, g^{r_{1,0}+r_{0,0}}\right)$. Note also that, we do not use a PKE scheme

to achieve the strong key-insulated security. Instead, an element $g' = g_2^{\alpha'}$ is included in the user secret key, and the public parameter g_1 is accordingly changed to be $g_1 = g^{\alpha'+\alpha_1+\alpha_0}$. Now, even if an adversary corrupts both the helper keys, security of all the periods are still ensured, since g' is unknown to the adversary. Figure 5 gives our detailed construction:

$\mathsf{KeyGen}(\lambda)$: Given a security parameter λ,

1. pick $\alpha', \alpha_0, \alpha_1 \xleftarrow{\$} \mathbb{Z}_p^*$, $g, g_2, h \xleftarrow{\$} \mathbb{G}$, and set $g' = g_2^{\alpha'}, g_1 = g^{\alpha'+\alpha_0+\alpha_1}$, $\mathsf{mst}_1 = g_2^{\alpha_1}$ and $\mathsf{mst}_0 = g_2^{\alpha_0}$,
2. pick a target collision-resistant hash function $H : \mathbb{Z} \to \mathbb{Z}_p^*$,
3. pick $r_{1,0}, r_{0,0}, r_{0,1} \xleftarrow{\$} \mathbb{Z}_p^*$,
4. set $(a_{0,0}, b_{0,0}) = \left(\mathsf{mst}_1\mathsf{mst}_0 \cdot (g_1^{H(0)}h)^{r_{1,0}+r_{0,0}}, g^{r_{1,0}+r_{0,0}}\right)$, $(a_{0,1}, b_{0,1}) = \left(\mathsf{mst}_0 \cdot (g_1^{H(1)}h)^{r_{0,1}}, g^{r_{0,1}}\right)$,
5. return $\mathsf{pk} = (g, g_1, g_2, h)$, $\mathsf{usk}_0 = (g', (a_{0,0}, b_{0,0}), (a_{0,1}, b_{0,1}))$ and $(\mathsf{mst}_1, \mathsf{mst}_0)$.

$\Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \bmod 2})$: To generate the update key hsk_t with the matching helper key $\mathsf{mst}_{t \bmod 2}$,

1. let $i = t \mod 2$,
2. pick $r_{i,t}, r_{i,t+1} \xleftarrow{\$} \mathbb{Z}_p^*$,
3. set $(\hat{a}_{i,t}, \hat{b}_{i,t}) = \left(\mathsf{mst}_i \cdot (g_1^{H(t)}h)^{r_{i,t}}, g^{r_{i,t}}\right)$ and $(\hat{a}_{i,t+1}, \hat{b}_{i,t+1}) = \left(\mathsf{mst}_i \cdot (g_1^{H(t+1)}h)^{r_{i,t+1}}, g^{r_{i,t+1}}\right)$,
4. return $\mathsf{hsk}_t = ((\hat{a}_{i,t}, \hat{b}_{i,t}), (\hat{a}_{i,t+1}, \hat{b}_{i,t+1}))$.

$\mathsf{Update}(t, \mathsf{usk}_{t-1}, \mathsf{hsk}_t)$: In period t, to update the user secret key from usk_{t-1} to usk_t,

1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. parse usk_{t-1} as $(g', (a_{j,t-1}, b_{j,t-1}), (a_{j,t}, b_{j,t}))$, and hsk_t as $((\hat{a}_{i,t}, \hat{b}_{i,t}), (\hat{a}_{i,t+1}, \hat{b}_{i,t+1}))$
3. set $a_{i,t} = a_{j,t} \cdot \hat{a}_{i,t}$, $b_{i,t} = b_{j,t} \cdot \hat{b}_{i,t}$, $a_{i,t+1} = \hat{a}_{i,t+1}$ and $b_{i,t+1} = \hat{b}_{i,t+1}$,
4. return $\mathsf{usk}_t = (g', (a_{i,t}, b_{i,t}), (a_{i,t+1}, b_{i,t+1}))$.

$\mathsf{Enc}(\mathsf{pk}, t, m)$: In period t, to encrypt a message $m \in \mathbb{G}_T$ under public key pk,

1. pick $s \xleftarrow{\$} \mathbb{Z}_p^*$, and compute $C_1 = e(g_1, g_2)^s \cdot m$, $C_2 = g^s$, $C_3 = (g_1^{H(t)}h)^s$,
2. return $\mathtt{CT} = (C_1, C_2, C_3)$.

$\mathsf{Dec}(\mathsf{usk}_t, \mathtt{CT})$: To decrypt a ciphertext $\mathtt{CT}$ with user secret key usk_t,

1. let $i = t \mod 2$,
2. parse usk_t as $(g', (a_{i,t}, b_{i,t}), (a_{i,t+1}, b_{i,t+1}))$, and $\mathtt{CT}$ as (C_1, C_2, C_3),
3. return $m \leftarrow \dfrac{C_1 \cdot e(C_3, b_{i,t})}{e(C_2, g' \cdot a_{i,t})}$.

Figure 5: Our Concrete PKIE Scheme Based on Boneh-Boyen IBE

Theorem 4 *The above Boneh-Boyen-based* PKIE *scheme is strongly key-insulated secure under chosen-plaintext attack, assuming the DBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

Theorem 4 follows directly from the following Lemmas 3 and 4.

Lemma 3 *The Boneh-Boyen-based* PKIE *scheme presented in Figure 5 is* IND-KI-CPA *secure, assuming the DBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

Proof. Given a polynomial time adversary $\mathcal{A}$ who can break the IND-KI-CPA security of our Boneh-Boyen-based PKIE scheme with advantage ϵ, we indicate that we can construct a polynomial time algorithm $\mathcal{B}$ that can break the DBDH assumption in groups $(\mathbb{G}, \mathbb{G}_T)$ with advantage $\frac{\epsilon}{4N}$, where N denotes the total number of periods.

Given a DBDH instance $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$ with unknown $a, b, c \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathcal{B}$'s goal is to decide whether $Z = e(g, g)^{abc}$. $\mathcal{B}$ first tosses a coin $\mathcal{COIN} \xleftarrow{\$} \{0, 1\}$ to guess which kind of adversary $\mathcal{A}$ will be. If $\mathcal{COIN} = 0$, it expects to face a Type I adversary, while for $\mathcal{COIN} = 1$, it forecasts a Type II adversary. $\mathcal{B}$ also chooses an index $\ell \xleftarrow{\$} \{1, \cdots, N\}$ as a guess for the target time period to be attacked by $\mathcal{A}$.

For $\mathcal{COIN} = 0$, $\mathcal{B}$ acts as the challenger and provides the simulations for $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\mathsf{IND\text{-}KI\text{-}CPA}}_{\mathcal{A},\mathtt{PKIE}}$ as follows:

`Init Stage.` Let $i^* = \ell \mod 2$ and $j^* = (\ell - 1) \mod 2$. $\mathcal{B}$ picks $\alpha', \alpha_{j^*}, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $g_2 = g^b, g' = g_2^{\alpha'}, g_1 = g^{\alpha'+\alpha_{j^*}} g^a$ and $h = g_1^{-H(\ell)} g^{\mu}$. Then $\mathcal{B}$ gives $\mathsf{pk} = (g, g_1, g_2, h)$ to $\mathcal{A}$. Note that the helper keys are implicitly $\mathsf{mst}_{i^*} = g^{ab}$ and $\mathsf{mst}_{j^*} = g_2^{\alpha_{j^*}}$.

`Find Stage.` In this stage, $\mathcal{A}$ issues a series of queries, and $\mathcal{B}$ responds as follows:

- User secret key oracle $\mathcal{O}_{\mathrm{u}}(t)$: If $t = \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, let $i = t \mod 2$, $j = (t-1) \mod 2$, and $\mathcal{B}$ responds to this query according to the following two cases:
  - If $i = i^*$: Pick $r'_{i,t}, r_{j,t}, r'_{i,t+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{usk}_t = (g', (a_{i,t}, b_{i,t}), (a_{i,t+1}, b_{i,t+1}))$ to $\mathcal{A}$, where

$$a_{i,t} = (g^b)^{-(\alpha' + \frac{\mu}{H(t)-H(\ell)})} \left(g_1^{H(t)} h\right)^{r'_{i,t}+r_{j,t}}, b_{i,t} = (g^b)^{\frac{-1}{H(t)-H(\ell)}} g^{r'_{i,t}+r_{j,t}}, \quad (1)$$

$$a_{i,t+1} = (g^b)^{-(\alpha' + \alpha_j + \frac{\mu}{H(t+1)-H(\ell)})} \left(g_1^{H(t+1)} h\right)^{r'_{i,t+1}}, b_{i,t+1} = (g^b)^{\frac{-1}{H(t+1)-H(\ell)}} g^{r'_{i,t+1}}. \quad (2)$$

    Observe that the above usk_t has the correct form, since letting $r_{i,t} = \frac{-b}{H(t)-H(\ell)} + r'_{i,t}$ and $r_{i,t+1} = \frac{-b}{H(t+1)-H(\ell)} + r'_{i,t+1}$, we have

$$\begin{aligned}
a_{i,t} &= (g^b)^{-(\alpha' + \frac{\mu}{H(t)-H(\ell)})} \left(g_1^{H(t)} h\right)^{r'_{i,t}+r_{j,t}} \\
&= g^{ab} g^{-ab} (g^b)^{\alpha_j} \cdot (g^b)^{-(\alpha' + \alpha_j + \frac{\mu}{H(t)-H(\ell)})} \left(g_1^{H(t)} h\right)^{r'_{i,t}+r_{j,t}} \\
&= g^{ab} g^{-ab} g_2^{\alpha_j} \cdot (g^{(\alpha'+\alpha_j)(H(t)-H(\ell))} g^{\mu})^{\frac{-b}{H(t)-H(\ell)}} \left(g_1^{H(t)} h\right)^{r'_{i,t}+r_{j,t}} \\
&= g^{ab} g_2^{\alpha_j} \left((g^{\alpha'+\alpha_j} g^a)^{H(t)-H(\ell)} g^{\mu}\right)^{\frac{-b}{H(t)-H(\ell)}} \left(g_1^{H(t)} h\right)^{r'_{i,t}+r_{j,t}} \\
&= g^{ab} g_2^{\alpha_j} \left(g_1^{H(t)} \cdot (g_1^{-H(\ell)} g^{\mu})\right)^{\frac{-b}{H(t)-H(\ell)}} \left(g_1^{H(t)} h\right)^{r'_{i,t}+r_{j,t}} \\
&= \mathsf{mst}_i \mathsf{mst}_j \left(g_1^{H(t)} h\right)^{\frac{-b}{H(t)-H(\ell)}+r'_{i,t}+r_{j,t}} \\
&= \mathsf{mst}_i \mathsf{mst}_j \left(g_1^{H(t)} h\right)^{r_{i,t}+r_{j,t}}, \\
b_{i,t} &= (g^b)^{\frac{-1}{H(t)-H(\ell)}} g^{r'_{i,t}+r_{j,t}} = g^{\frac{-b}{H(t)-H(\ell)}+r'_{i,t}+r_{j,t}} = g^{r_{i,t}+r_{j,t}}, \\
a_{i,t+1} &= (g^b)^{-(\alpha' + \alpha_j + \frac{\mu}{H(t+1)-H(\ell)})} \cdot (g_1^{H(t+1)} h)^{r'_{i,t+1}} \\
&= g^{ab} g^{-ab} \cdot (g^{(\alpha'+\alpha_j)(H(t+1)-H(\ell))} g^{\mu})^{\frac{-b}{H(t+1)-H(\ell)}} \cdot (g_1^{H(t+1)} h)^{r'_{i,t+1}} \\
&= g^{ab} \cdot \left((g^{\alpha'+\alpha_j} g^a)^{H(t+1)-H(\ell)} g^{\mu}\right)^{\frac{-b}{H(t+1)-H(\ell)}} \cdot (g_1^{H(t+1)} h)^{r'_{i,t+1}} \\
&= g^{ab} \cdot \left(g_1^{H(t+1)} g_1^{-H(\ell)} g^{\mu}\right)^{\frac{-b}{H(t+1)-H(\ell)}} \cdot (g_1^{H(t+1)} h)^{r'_{i,t+1}} \\
&= g^{ab} \cdot \left(g_1^{H(t+1)} h\right)^{\frac{-b}{H(t+1)-H(\ell)}+r'_{i,t+1}} \\
&= \mathsf{mst}_i \cdot \left(g_1^{H(t+1)} h\right)^{r_{i,t+1}}, \\
b_{i,t+1} &= (g^b)^{\frac{-1}{H(t+1)-H(\ell)}} g^{r'_{i,t+1}} = g^{\frac{-b}{H(t+1)-H(\ell)}+r'_{i,t+1}} = g^{r_{i,t+1}}.
\end{aligned}$$

  - If $j = i^*$: Pick $r_{i,t}, r'_{j,t}, r_{i,t+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{usk}_t = (g', (a_{i,t}, b_{i,t}), (a_{i,t+1}, b_{i,t+1}))$ to $\mathcal{A}$, where

$$a_{i,t} = (g^b)^{-(\alpha' + \frac{\mu}{H(t)-H(\ell)})} \cdot (g_1^{H(t)} h)^{r_{i,t}+r'_{j,t}}, \quad b_{i,t} = (g^b)^{\frac{-1}{H(t)-H(\ell)}} g^{r_{i,t}+r'_{j,t}}, \quad (3)$$

$$a_{i,t+1} = g_2^{\alpha_i} (g_1^{H(t+1)} h)^{r_{i,t+1}}, \quad b_{i,t+1} = g^{r_{i,t+1}}. \quad (4)$$

    Letting $r_{j,t} = \frac{-b}{H(t)-H(\ell)} + r'_{j,t}$, we can similarly see that the above usk_t has the correct form as required.
- Helper key query $\mathcal{O}_{\mathrm{h}}(i)$ with $i \in \{0,1\}$: $\mathcal{B}$ outputs a random bit and **aborts**, since it means that $\mathcal{B}$ guessed the wrong $\mathcal{COIN}$.

`Challenge.` Once $\mathcal{A}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathbb{G}_T$ on which it wishes to be challenged. If $t^* \neq \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ picks $\beta \xleftarrow{\$} \{0,1\}$, and gives the challenge ciphertext $\mathtt{CT}^* = (C_1^*, C_2^*, C_3^*) = (Z \cdot e(g^c, g^b)^{\alpha' + \alpha_{j^*}} \cdot m_\beta, g^c, (g^c)^\mu)$ to $\mathcal{A}$.

Observe that, if $Z = e(g,g)^{abc}$, the above challenge ciphertext has the correct form as required, since

$$\begin{aligned} C_1^* &= Z \cdot e(g^c, g^b)^{\alpha' + \alpha_{j^*}} \cdot m_\beta = e(g,g)^{abc} \cdot e(g^{\alpha' + \alpha_{j^*}}, g^b)^c \cdot m_\beta \\ &= e(g^{\alpha' + \alpha_{j^*}} g^a, g^b)^c \cdot m_\beta = e(g_1, g_2)^c \cdot m_\beta, \\ C_3^* &= (g^c)^\mu = (g^\mu)^c = \left(g_1^{H(\ell)} g_1^{-H(\ell)} g^\mu\right)^c = \left(g_1^{H(\ell)} h\right)^c = \left(g_1^{H(t^*)} h\right)^c \end{aligned}$$

On the other hand, when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext $\mathtt{CT}^*$ is independent of β in the adversary's view.

`Guess stage.` $\mathcal{A}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{B}$ also responds as in find stage.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta' = \beta$, then $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

If $\mathcal{COIN} = 1$, $\mathcal{B}$ provides the simulation for $\mathcal{A}$ as follows:

`Init Stage.` $\mathcal{B}$ first picks a random bit $\nu \xleftarrow{\$} \{0,1\}$ to guess that $\mathcal{A}$ will corrupt the helper key $\mathsf{mst}_{\overline{\nu}}$, where $\overline{\nu}$ denotes $1 - \nu$. $\mathcal{B}$ picks $\alpha', \alpha_{\overline{\nu}}, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $g_2 = g^b, g' = g_2^{\alpha'}, g_1 = g^{\alpha' + \alpha_{\overline{\nu}}} g^a$ and $h = g_1^{-H(\ell)} g^\mu$. Then $\mathcal{B}$ gives $\mathsf{pk} = (g, g_1, g_2, h)$ to $\mathcal{A}$. Note that the helper keys are implicitly $\mathsf{mst}_\nu = g^{ab}$ and $\mathsf{mst}_{\overline{\nu}} = g_2^{\alpha_{j^*}}$, where mst_ν is unknown to $\mathcal{B}$.

`Find Stage.` In this stage, $\mathcal{A}$ issues a series of queries, and $\mathcal{B}$ responds as follows:

- User secret key oracle $\mathcal{O}_{\mathrm{u}}(t)$: $\mathcal{B}$ outputs a random bit and **aborts** if $t = \ell$ or $(t = \ell - 1) \wedge (\overline{\nu} = \ell \mod 2)$. Otherwise, let $i = t \mod 2$, $j = (t-1) \mod 2$, and $\mathcal{B}$ responds to this query according to the following two cases:
  - If $i = \nu$: Pick $r'_{i,t}, r_{j,t}, r'_{i,t+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{usk}_t = (g', (a_{i,t}, b_{i,t}), (a_{i,t+1}, b_{i,t+1}))$ to $\mathcal{A}$, where $a_{i,t}, b_{i,t}, a_{i,t+1}$ and $b_{i,t+1}$ are generated as in Eqs. (1) and (2).
  - If $i = \overline{\nu}$: Pick $r_{i,t}, r'_{j,t}, r_{i,t+1} \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{usk}_t = (g', (a_{i,t}, b_{i,t}), (a_{i,t+1}, b_{i,t+1}))$ to $\mathcal{A}$, where $a_{i,t}, b_{i,t}, a_{i,t+1}$ and $b_{i,t+1}$ are generated as in Eqs. (3) and (4).
- Helper key query $\mathcal{O}_{\mathrm{h}}(i)$ with $i \in \{0,1\}$: If $i = \nu$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ returns $\mathsf{mst}_{\overline{\nu}}$ to $\mathcal{A}$.

`Challenge.` Once $\mathcal{A}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathbb{G}_T$ on which it wishes to be challenged. If $t^* \neq \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ picks $\beta \xleftarrow{\$} \{0,1\}$, and gives the challenge ciphertext $\mathtt{CT}^* = (C_1^*, C_2^*, C_3^*) = (Z \cdot e(g^c, g^b)^{\alpha' + \alpha_{\overline{\nu}}} \cdot m_\beta, g^c, (g^c)^\mu)$ to $\mathcal{A}$. It can also verified that, if $Z = e(g,g)^{abc}$ the challenge ciphertext $\mathtt{CT}^*$ has the correct form as required, and when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext $\mathtt{CT}^*$ is independent of β in the adversary's view.

`Guess stage.` $\mathcal{A}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{B}$ also responds as in find stage.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta' = \beta$, then $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

Analysis: The simulations provided for $\mathcal{A}$ are perfect, unless the following events happens:

E_1: a helper key query $\mathcal{O}_{\mathrm{h}}(i)$ was issued when $\mathcal{COIN} = 0$;
E_2: the helper key query $\mathcal{O}_{\mathrm{h}}(\nu)$ was issued when $\mathcal{COIN} = 1$;
E_3: the user secret key query $\mathcal{O}_{\mathrm{u}}(\ell)$ was issued when $\mathcal{COIN} = 0$;
E_4: the user secret key query $\mathcal{O}_{\mathrm{u}}(t)$ was issued such that $t = \ell$ or $(t = \ell - 1) \wedge (\overline{\nu} = \ell \bmod 2)$ when $\mathcal{COIN} = 1$;

To analyze the above events, we alternately consider the following events:

H_1: $\mathcal{B}$ successfully guesses $\ell = t^*$;
H_2: $\mathcal{B}$ successfully guesses the kind of attack produced by $\mathcal{A}$;
H_3: $\mathcal{B}$ luckily predicts which helper's key is exposed when $\mathcal{COIN} = 1$

Clearly, we have $\Pr[H_1] = 1/N$, $\Pr[H_2] = 1/2$ and $\Pr[H_3] = 1/2$. Also, we have $H_1 \Rightarrow \neg E3, H_2 \Rightarrow \neg E1, H_2 \wedge H_3 \Rightarrow \neg E4$ and $H_3 \Rightarrow \neg E2$. The conjunction of events H_1, H_2 and H_3 is readily seen to occur with probability greater than $1/4N$, and hence $\mathcal{B}$'s advantage is greater than $\epsilon/4N$. It is also clear that $\mathcal{B}$'s running time is polynomially bounded. Thus the proof of Lemma 3 is concluded. □

Lemma 4 *The Boneh-Boyen-based* PKIE *scheme presented in Figure 5 is* strong-IND-KI-CPA *secure, assuming the DBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

Proof. Given a polynomial time adversary $\mathcal{A}$ who can break the strong-IND-KI-CPA security of our Boneh-Boyen-based PKIE scheme with advantage ϵ, we indicate that we can construct a polynomial time algorithm $\mathcal{B}$ that can break the DBDH assumption in groups $(\mathbb{G}, \mathbb{G}_T)$ with advantage ϵ/N, where N denotes the total number of periods.

Given a DBDH instance $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$ with unknown $a, b, c \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathcal{B}$'s goal is to decide whether $Z = e(g,g)^{abc}$. $\mathcal{B}$ acts as the challenger and provides the simulations for $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\text{strong-IND-KI-CPA}}_{\mathcal{A},\mathtt{PKIE}}$ as follows:

`Init Stage.` $\mathcal{B}$ first chooses an index $\ell \xleftarrow{\$} \{1, \cdots, N\}$ as a guess for the target time period to be attacked by $\mathcal{A}$. Next, $\mathcal{B}$ picks $\alpha_0, \alpha_1, \mu \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $g_1 = g^a g^{\alpha_0 + \alpha_1}, g_2 = g^b$, $h = g_1^{-H(\ell)} g^{\mu}$, $\mathsf{mst}_1 = g_2^{\alpha_1}$ and $\mathsf{mst}_0 = g_2^{\alpha_0}$. Finally, $\mathcal{B}$ gives $\mathsf{pk} = (g, g_1, g_2, h)$ and $(\mathsf{mst}_1, \mathsf{mst}_0)$ to $\mathcal{A}$. Note that g' is implicitly set to be $g' = g^{ab}$ which is unknown to $\mathcal{B}$.

`Challenge.` $\mathcal{A}$ outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathbb{G}_T$ on which it wishes to be challenged. If $t^* \neq \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ picks $\beta \xleftarrow{\$} \{0,1\}$, and gives the challenge ciphertext $\mathtt{CT}^* = (C_1^*, C_2^*, C_3^*) = (Z \cdot e(g^b, g^c)^{\alpha_0 + \alpha_1} \cdot m_\beta, g^c, (g^c)^{\mu})$ to $\mathcal{A}$. It is readily to see that, if $Z = e(g,g)^{abc}$ the above challenge ciphertext has the correct form as required, and when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext $\mathtt{CT}^*$ is independent of β in the adversary's view.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta' = \beta$, then $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

Analysis: The simulations provided for $\mathcal{A}$ are perfect, if $\mathcal{B}$ correctly guesses $\ell = t^*$, whose probability is obviously $1/N$. Thus $\mathcal{B}$'s advantage against the DBDH assumption is at least ϵ/N. It is also clear that $\mathcal{B}$'s running time is polynomially bounded. Thus the proof of Lemma 4 is concluded. □

Remark 6. Like LQY scheme, since our PKIE scheme are based on the selective-ID secure Boneh-Boyen IBE scheme, the security bound of our scheme will be degraded by a factor of the total number of periods. Recently, Lewko and Waters [34] proposed a fully scheme (H)IBE scheme with short ciphertexts. Based on Lewko-Waters IBE, our scheme can avoid this security degradation.

4.2.2 Efficient Instantiation from Boneh-Boyen-Goh 2-HIBE

Based on Boneh-Boyen-Goh two-level HIBE scheme [7] which is reviewed in Figure 6, we here present an HIBE-based instantiation of PKIE scheme. The detailed scheme is given in Figure 7. Our scheme consists of two HIBE systems. To improve the efficiency, we do not use the PKE scheme to achieve the strong key-insulated security, instead, we introduce an element $g' = g_2^{\alpha'}$ into the user secret key. In addition, we further shorten the public key, by sharing the public parameters (g, g_2, f) and integrating the parameters $(g^{\alpha'}, g^{\alpha_1}, g^{\alpha_0})$ into a single $g^{\alpha'+\alpha_1+\alpha_0}$. The resulting PKIE scheme is comparable with LQY scheme: the same public key size, ciphertext size, encryption cost and decryption cost, only with slight longer user secret keys and heavier cost for user key-updates.

<table>
<tr><td colspan="2">$\mathsf{Setup}(\lambda)$:
1. pick $\alpha \xleftarrow{\$} \mathbb{Z}_p^*$, $g, g_2, f, h \xleftarrow{\$} \mathbb{G}$, and set $g_1 = g^\alpha$,
2. output $param = (g, g_1, g_2, f, h)$ and $msk = g_2^\alpha$.</td></tr>
<tr><td>$\mathsf{Extract}(param, msk, \mathsf{ID}_1)$:
1. pick $r \xleftarrow{\$} \mathbb{Z}_p^*$,
2. set $a_{\mathsf{ID}_1} = msk \cdot (g_1^{\mathsf{ID}_1} h)^r$, $b_{\mathsf{ID}_1} = g^r$ and $c_{\mathsf{ID}_1} = f^r$,
3. output $sk_{\mathsf{ID}_1} = (a_{\mathsf{ID}_1}, b_{\mathsf{ID}_1}, c_{\mathsf{ID}_1})$.</td><td>or $\mathsf{Extract}(param, sk_{\mathsf{ID}_1}, \mathsf{ID} = (\mathsf{ID}_1, \mathsf{ID}_2))$:
1. parse sk_{ID_1} as $(a_{\mathsf{ID}_1}, b_{\mathsf{ID}_1}, c_{\mathsf{ID}_1})$,
2. pick $r' \xleftarrow{\$} \mathbb{Z}_p^*$,
3. set $a_{\mathsf{ID}} = a_{\mathsf{ID}_1} \cdot c_{\mathsf{ID}_1}^{\mathsf{ID}_2} \cdot (g_1^{\mathsf{ID}_1} f^{\mathsf{ID}_2} h)^{r'}$ and $b_{\mathsf{ID}} = b_{\mathsf{ID}_1} \cdot g^{r'}$,
4. output $sk_{\mathsf{ID}} = (a_{\mathsf{ID}}, b_{\mathsf{ID}})$.</td></tr>
<tr><td>$\mathsf{Enc}(param, \mathsf{ID}, m)$:
1. pick $s \xleftarrow{\$} \mathbb{Z}_p^*$, and set $C_1 = e(g_1, g_2)^s \cdot m, C_2 = g^s$,
2. if ID is a 2-level identity $(\mathsf{ID}_1, \mathsf{ID}_2)$, then set $C_3 = (g_1^{\mathsf{ID}_1} f^{\mathsf{ID}_2} h)^s$; else set $C_3 = (g_1^{\mathsf{ID}} h)^s$,
3. return $C = (C_1, C_2, C_3)$.</td><td>$\mathsf{Dec}(sk_{\mathsf{ID}}, C)$:
1. parse C as (C_1, C_2, C_3)
2. parse sk_{ID} as $(a_{\mathsf{ID}}, b_{\mathsf{ID}})$ or $(a_{\mathsf{ID}}, b_{\mathsf{ID}}, c_{\mathsf{ID}})$,
3. return $m \leftarrow \dfrac{C_1 \cdot e(C_3, b_{\mathsf{ID}})}{e(C_2, a_{\mathsf{ID}})}$.</td></tr>
</table>

Figure 6: Boneh-Boyen-Goh 2-HIBE Scheme [7]

<table>
<tr><td colspan="2">$\mathsf{KeyGen}(\lambda)$: Given a security parameter λ,
1. pick a collision-resistant hash function $H : \mathbb{Z} \to \mathbb{Z}_p^*$,
2. pick $g, g_2, f, h_0, h_1 \xleftarrow{\$} \mathbb{G}$ and $\alpha_0, \alpha_1, \alpha' \xleftarrow{\$} \mathbb{Z}_p^*$, and set $g_1 = g^{\alpha'+\alpha_0+\alpha_1}$, $g' = g_2^{\alpha'}$, $\mathsf{mst}_1 = g_2^{\alpha_1}$, $\mathsf{mst}_0 = g_2^{\alpha_0}$,
3. pick $r_{1,0}, r_{0,0} \xleftarrow{\$} \mathbb{Z}_p^*$, and set $(a_{1,0}, b_{1,0}) = \left(\mathsf{mst}_1 \cdot (g_1^{H(-1)} f^{H(0)} h_1)^{r_{1,0}}, g^{r_{1,0}}\right)$ and $(a_{0,0}, b_{0,0}, c_{0,0}) = \left(\mathsf{mst}_0 \cdot (g_1^{H(0)} h_0)^{r_{0,0}}, g^{r_{0,0}}, f^{r_{0,0}}\right)$,
4. return $\mathsf{pk} = (g, g_1, g_2, f, h_0, h_1)$, $\mathsf{usk}_0 = (g', (a_{1,0}, b_{1,0}), (a_{0,0}, b_{0,0}, c_{0,0}))$, mst_1 and mst_0.</td></tr>
<tr><td>$\Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \bmod 2})$: To generate the update key hsk_t with the matching helper key $\mathsf{mst}_{t \bmod 2}$,
1. let $i = t \mod 2$,
2. pick $r_{i,t} \xleftarrow{\$} \mathbb{Z}_p^*$,
3. set $(a_{i,t}, b_{i,t}, c_{i,t}) = \left(\mathsf{mst}_i \cdot (g_1^{H(t)} h_i)^{r_{i,t}}, g^{r_{i,t}}, f^{r_{i,t}}\right)$,
4. return $\mathsf{hsk}_t = (a_{i,t}, b_{i,t}, c_{i,t})$.</td><td>$\mathsf{Update}(t, \mathsf{usk}_{t-1}, \mathsf{hsk}_t)$: In period t, to update the user secret key from usk_{t-1} to usk_t,
1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. parse hsk_t as $(a_{i,t}, b_{i,t}, c_{i,t})$, and usk_{t-1} as $(g', (a_{i,t-1}, b_{i,t-1}), (a_{j,t-1}, b_{j,t-1}, c_{j,t-1}))$,
3. pick $r_{j,t} \xleftarrow{\$} \mathbb{Z}_p^*$,
4. set $a_{j,t} = a_{j,t-1} \cdot c_{j,t-1}^{H(t)} \cdot (g_1^{H(t-1)} f^{H(t)} h_j)^{r_{j,t}}$ and $b_{j,t} = b_{j,t-1} \cdot g^{r_{j,t}}$,
5. return $\mathsf{usk}_t = (g', (a_{j,t}, b_{j,t}), (a_{i,t}, b_{i,t}, c_{i,t}))$.</td></tr>
<tr><td>$\mathsf{Enc}(\mathsf{pk}, t, m)$: In period t, to encrypt a message m under public key pk,
1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. pick $s \xleftarrow{\$} \mathbb{Z}_p^*$, and set $C_1 = e(g_1, g_2)^s \cdot m$, $C_2 = g^s$, $C_3 = (g_1^{H(t-1)} f^{H(t)} h_j)^s$ and $C_4 = (g_1^{H(t)} h_i)^s$,
3. return $\mathtt{CT} = (C_1, C_2, C_3, C_4)$.</td><td>$\mathsf{Dec}(\mathsf{usk}_t, \mathtt{CT})$: To decrypt a ciphertext $\mathtt{CT}$ with user secret key usk_t,
1. let $i = t \mod 2$ and $j = (t-1) \mod 2$,
2. parse $\mathtt{CT}$ as (C_1, C_2, C_3, C_4), and usk_t as $(g', (a_{j,t}, b_{j,t}), (a_{i,t}, b_{i,t}, c_{i,t}))$,
3. return $m \leftarrow \dfrac{C_1 \cdot e(b_{j,t}, C_3) \cdot e(b_{i,t}, C_4)}{e\left(C_2, g' a_{j,t} a_{i,t}\right)}$.</td></tr>
</table>

Figure 7: Our Concrete PKIE Scheme Based on Boneh-Boyen-Goh 2-HIBE

Theorem 5 *The above Boneh-Boyen-Goh-based* PKIE *scheme is strongly key-insulated secure*

under chosen-plaintext attack, assuming the 2-DBDHE assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$.

Theorem 5 follows directly from the following Lemmas 5 and 6.

Lemma 5 *The Boneh-Boyen-Goh-based* PKIE *scheme presented in Figure 7 is* IND-KI-CPA *secure, assuming the 2-DBDHE assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

Proof. Given a polynomial time adversary $\mathcal{A}$ who can break the IND-KI-CPA security of our Boneh-Boyen-Goh-based PKIE scheme with advantage ϵ, we indicate that we can construct a polynomial time algorithm $\mathcal{B}$ that can break the 2-DBDHE assumption in groups $(\mathbb{G}, \mathbb{G}_T)$ with advantage $\frac{\epsilon}{4N}$, where N denotes the total number of periods.

Given a 2-DBDHE instance $(g, h, g^a, g^{a^2}, g^{a^4}, Z) \in \mathbb{G}^5 \times \mathbb{G}_T$ with unknown $a \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathcal{B}$'s goal is to decide whether $Z = e(g,h)^{a^3}$. $\mathcal{B}$ first tosses a coin $\mathcal{COIN} \xleftarrow{\$} \{0,1\}$ to guess which kind of adversary $\mathcal{A}$ will be. If $\mathcal{COIN} = 0$, it expects to face a Type I adversary, while for $\mathcal{COIN} = 1$, it forecasts a Type II adversary. $\mathcal{B}$ also chooses an index $\ell \xleftarrow{\$} \{1, \cdots, N\}$ as a guess for the target time period to be attacked by $\mathcal{A}$.

For $\mathcal{COIN} = 0$, $\mathcal{B}$ acts as the challenger and provides the simulations for $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\text{IND-KI-CPA}}_{\mathcal{A},\text{PKIE}}$ as follows:

`Init Stage.` Let $i^* = \ell \mod 2$ and $j^* = (\ell - 1) \mod 2$. $\mathcal{B}$ picks $\alpha', \alpha_{j^*}, \gamma, \theta, \omega_{i^*}, \omega_{j^*} \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $g_1 = g^{\alpha' + \alpha_{j^*}} g^a, g_2 = g^{a^2} g^\gamma, g' = g_2^{\alpha'}, f = g^{a^2} g^\theta, h_{i^*} = g_1^{-H(\ell)} g^{\omega_{i^*}}, h_{j^*} = g_1^{-H(\ell-1)} f^{-H(\ell)} g^{\omega_{j^*}}$. Then $\mathcal{B}$ gives $\mathsf{pk} = (g, g_1, g_2, f, h_0, h_1)$ to $\mathcal{A}$. Note that the helper keys are implicitly $\mathsf{mst}_{i^*} = g^{a^3} g^{a \cdot \gamma}$ and $\mathsf{mst}_{j^*} = g_2^{\alpha_{j^*}}$. Here mst_{i^*} is unknown to $\mathcal{B}$.

`Find Stage.` In this stage, $\mathcal{A}$ issues a series of queries, and $\mathcal{B}$ responds as follows:

- User secret key oracle $\mathcal{O}_{\text{u}}(t)$: If $t = \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, letting $i = t \mod 2$ and $j = (t-1) \mod 2$, $\mathcal{B}$ responds to this query according to the following two cases:
  - If $i = i^*$: Pick $r'_{i,t}, r_{j,t} \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{usk}_t = (g', (a_{j,t}, b_{j,t}), (a_{i,t}, b_{i,t}, c_{i,t}))$ to $\mathcal{A}$, where

$$a_{j,t} = \mathsf{mst}_j \cdot \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r_{j,t}}, \quad b_{j,t} = g^{r_{j,t}}, \tag{5}$$

$$a_{i,t} = g^{a \cdot \gamma} \cdot \left(g^{a^2}\right)^{-(\alpha' + \alpha_j + \frac{\omega_i}{H(t) - H(\ell)})} \cdot \left(g_1^{H(t)} h_i\right)^{r'_{i,t}}, \; b_{i,t} = \left(g^{a^2}\right)^{\frac{-1}{H(t)-H(\ell)}} g^{r'_{i,t}} \tag{6}$$

$$c_{i,t} = \left(g^{a^4}\right)^{\frac{-1}{H(t)-H(\ell)}} \left(g^{a^2}\right)^{r'_{i,t} - \frac{\theta}{H(t)-H(\ell)}} g^{\theta \cdot r'_{i,t}}. \tag{7}$$

Observe that the above usk_t has the correct form, since if let $r_{i,t} = \frac{-a^2}{H(t)-H(\ell)} + r'_{i,t}$,

we have

$$
\begin{aligned}
a_{i,t} &= g^{a\cdot\gamma}\cdot\left(g^{a^2}\right)^{-(\alpha'+\alpha_j+\frac{\omega_i}{H(t)-H(\ell)})}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g^{a\cdot\gamma}\cdot\left(g^{-a^2}\right)^{\alpha'+\alpha_j}\left(g^{a^2}\right)^{\frac{-\omega_i}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g^{a^3}g^{-a^3}g^{a\cdot\gamma}\cdot\left(g^{-a^2}\right)^{\alpha'+\alpha_j}\left(g^{a^2}\right)^{\frac{-\omega_i}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g^{a^3}g^{a\cdot\gamma}\cdot(g^{\alpha'+\alpha_j}g^{a})^{-a^2}\left(g^{a^2}\right)^{\frac{-\omega_i}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g^{a^3}g^{a\cdot\gamma}g_1^{-a^2}\,(g^{\omega_i})^{\frac{-a^2}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g^{a^3}g^{a\cdot\gamma}\cdot\left(g_1^{H(t)-H(\ell)}g^{\omega_i}\right)^{\frac{-a^2}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g^{a^3}g^{a\cdot\gamma}\cdot\left(g_1^{H(t)}g_1^{-H(\ell)}g^{\omega_i}\right)^{\frac{-a^2}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= \left(g^{a^2}g^{\gamma}\right)^{a}\cdot\left(g_1^{H(t)}h_i\right)^{\frac{-a^2}{H(t)-H(\ell)}}\cdot\left(g_1^{H(t)}h_i\right)^{r'_{i,t}}\\
&= g_2^{\alpha_i}\cdot\left(g_1^{H(t)}h_i\right)^{\frac{-a^2}{H(t)-H(\ell)}+r'_{i,t}}\\
&= \mathsf{mst}_i\cdot\left(g_1^{H(t)}h_i\right)^{r_{i,t}},\\
b_{i,t} &= \left(g^{a^2}\right)^{\frac{-1}{H(t)-H(\ell)}}g^{r'_{i,t}} = g^{\frac{-a^2}{H(t)-H(\ell)}+r'_{i,t}} = g^{r_{i,t}},\\
c_{i,t} &= \left(g^{a^4}\right)^{\frac{-1}{H(t)-H(\ell)}}\left(g^{a^2}\right)^{r'_{i,t}-\frac{\theta}{H(t)-H(\ell)}}g^{\theta\cdot r'_{i,t}}\\
&= \left(g^{a^2}\right)^{\frac{-a^2}{H(t)-H(\ell)}}\left(g^{a^2}\right)^{r'_{i,t}}(g^{\theta})^{\frac{-a^2}{H(t)-H(\ell)}+r'_{i,t}}\\
&= \left(g^{a^2}\right)^{\frac{-a^2}{H(t)-H(\ell)}+r'_{i,t}}(g^{\theta})^{\frac{-a^2}{H(t)-H(\ell)}+r'_{i,t}} = \left(g^{a^2}g^{\theta}\right)^{\frac{-a^2}{H(t)-H(\ell)}+r'_{i,t}} = f^{r_{i,t}}.
\end{aligned}
$$

– If $j = i^*$: Pick $r_{i,t}, r'_{j,t} \xleftarrow{\$} \mathbb{Z}_p^*$, and returns $\mathsf{usk}_t = (g', (a_{j,t}, b_{j,t}), (a_{i,t}, b_{i,t}, c_{i,t}))$ to $\mathcal{A}$, where

$$a_{j,t} = (g^a)^{\gamma-\theta-\frac{(\alpha'+\alpha_i)(H(t-1)-H(\ell))+\omega_j}{H(t)}}\left(g^{a^2}\right)^{\frac{H(\ell)-H(t-1)}{H(t)}}\left(g_1^{H(t-1)}f^{H(t)}h_j\right)^{r'_{j,t}}, \quad (8)$$

$$b_{j,t} = (g^a)^{\frac{-1}{H(t)}}g^{r'_{j,t}}, \quad (9)$$

$$a_{i,t} = \mathsf{mst}_i\cdot\left(g_1^{H(t)}h_i\right)^{r_{i,t}}, \quad b_{i,t} = g^{r_{i,t}},\ c_{i,t} = f^{r_{i,t}}. \quad (10)$$

Letting $r_{j,t} = \frac{-a}{H(t)} + r'_{j,t}$, we can see that the above usk_t has the correct form as

required, since

$$
\begin{aligned}
a_{j,t} &= (g^a)^{\gamma-\theta-\frac{(\alpha'+\alpha_i)(H(t-1)-H(\ell))+\omega_j}{H(t)}} \cdot \left(g^{a^2}\right)^{\frac{H(\ell)-H(t-1)}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= (g^a)^{\gamma-\theta} \cdot \left(g^{(\alpha'+\alpha_i)(H(t-1)-H(\ell))+\omega_j} (g^a)^{H(t-1)-H(\ell)}\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= g^{a^3} g^{-a^3} (g^a)^{\gamma-\theta} \cdot \left((g^{\alpha'+\alpha_i} g^a)^{H(t-1)-H(\ell)} g^{\omega_j}\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= g^{a^3} g^{a\cdot\gamma} \left(g^{a^2} g^{\theta}\right)^{-a} \cdot \left(g_1^{H(t-1)-H(\ell)} g^{\omega_j}\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= g^{a^3} g^{a\cdot\gamma} f^{-a} \cdot \left(g_1^{H(t-1)-H(\ell)} g^{\omega_j}\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= g^{a^3} g^{a\cdot\gamma} \cdot \left(g_1^{H(t-1)-H(\ell)} f^{H(t)} g^{\omega_j}\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= \left(g^{a^2} g^{\gamma}\right)^{a} \cdot \left(g_1^{H(t-1)} f^{H(t)} g_1^{-H(\ell)} g^{\omega_j}\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= \left(g^{a^2} g^{\gamma}\right)^{a} \cdot \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{\frac{-a}{H(t)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}} \\
&= \left(g^{a^2} g^{\gamma}\right)^{a} \cdot \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{\frac{-a}{H(t)}+r'_{j,t}} \\
&= \mathsf{mst}_j \cdot \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r_{j,t}}, \\
b_{j,t} &= (g^a)^{\frac{-1}{H(t)}} g^{r'_{j,t}} = g^{\frac{-a}{H(t)}+r'_{j,t}} = g^{r_{j,t}}.
\end{aligned}
$$

- Helper key query $\mathcal{O}_{\mathrm{h}}(i)$ with $i \in \{0,1\}$: $\mathcal{B}$ outputs a random bit and **aborts**, since it means that $\mathcal{B}$ guessed the wrong $\mathcal{COIN}$.

`Challenge.` Once $\mathcal{A}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathbb{G}_T$ on which it wishes to be challenged. If $t^* \neq \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ picks $\beta \xleftarrow{\$} \{0,1\}$, and gives the challenge ciphertext $\mathtt{CT}^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ to $\mathcal{A}$, where

$$
C_1^* = Z \cdot e\left(h, (g^a)^{\gamma} (g^{a^2} g^{\gamma})^{\alpha'+\alpha_{j^*}}\right) \cdot m_\beta, \quad C_2^* = h, \quad C_3^* = h^{\omega_{j^*}}, \quad C_4^* = h^{\omega_{i^*}}.
$$

Observe that, if $Z = e(g,h)^{a^3}$, the above challenge ciphertext has the correct form as required, since letting $h = g^c$, we have

$$
\begin{aligned}
C_1^* &= Z \cdot e\left(h, (g^a)^{\gamma} (g^{a^2} g^{\gamma})^{\alpha'+\alpha_{j^*}}\right) \cdot m_\beta \\
&= e(g,g^c)^{a^3} \cdot e\left(g^c, (g^a)^{\gamma} (g^{a^2} g^{\gamma})^{\alpha'+\alpha_{j^*}}\right) \cdot m_\beta \\
&= e(g^a, g^{a^2})^c \cdot e(g^c, (g^a)^{\gamma}) \cdot e(g^c, (g^{a^2} g^{\gamma})^{\alpha'+\alpha_{j^*}}) \cdot m_\beta \\
&= e(g^a, g^{a^2})^c \cdot e(g^a, g^{\gamma})^c \cdot e(g^{\alpha'+\alpha_{j^*}}, g^{a^2} g^{\gamma})^c \cdot m_\beta \\
&= e(g^a, g^{a^2} g^{\gamma})^c \cdot e(g^{\alpha'+\alpha_{j^*}}, g^{a^2} g^{\gamma})^c \cdot m_\beta \\
&= e(g^a g^{\alpha'+\alpha_{j^*}}, g^{a^2} g^{\gamma})^c \cdot m_\beta \\
&= e(g_1, g_2)^c \cdot m_\beta, \\
C_2^* &= h = g^c, \\
C_3^* &= h^{\omega_{j^*}} = (g^{\omega_{j^*}})^c = \left(g_1^{H(\ell-1)} f^{H(\ell)} g_1^{-H(\ell-1)} f^{-H(\ell)} g^{\omega_{j^*}}\right)^c = \left(g_1^{H(t^*-1)} f^{H(t^*)} h_{j^*}\right)^c, \\
C_4^* &= h^{\omega_{i^*}} = (g^{\omega_{i^*}})^c = \left(g_1^{H(\ell)} g_1^{-H(\ell)} g^{\omega_{i^*}}\right)^c = \left(g_1^{H(\ell)} h_{i^*}\right)^c == \left(g_1^{H(t^*)} h_{i^*}\right)^c.
\end{aligned}
$$

On the other hand, when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext CT^* is independent of β in the adversary's view.

`Guess stage.` $\mathcal{A}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{B}$ also responds as in find stage.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta' = \beta$, then $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

If $\mathcal{COIN} = 1$, $\mathcal{B}$ provides the simulation for $\mathcal{A}$ as follows:

`Init Stage.` $\mathcal{B}$ first picks a random bit $\nu \xleftarrow{\$} \{0,1\}$ to guess that $\mathcal{A}$ will corrupt the helper key $\mathsf{mst}_{\overline{\nu}}$, where $\overline{\nu}$ denotes $1-\nu$. $\mathcal{B}$ picks $\alpha', \alpha_{\overline{\nu}}, \gamma, \theta, \omega_0, \omega_1 \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $g_1 = g^{\alpha'+\alpha_{\overline{\nu}}} g^a, g_2 = g^{a^2} g^{\gamma}, g' = g_2^{\alpha'}, f = g^{a^2} g^{\theta}$. If $\nu = \ell \mod 2$, $\mathcal{B}$ defines $h_\nu = g_1^{-H(\ell)} g^{\omega_\nu}$ and $h_{\overline{\nu}} = g_1^{-H(\ell-1)} f^{-H(\ell)} g^{\omega_{\overline{\nu}}}$. Otherwise, $\mathcal{B}$ defines $h_{\overline{\nu}} = g_1^{-H(\ell)} g^{\omega_{\overline{\nu}}}$ and $h_\nu = g_1^{-H(\ell-1)} f^{-H(\ell)} g^{\omega_\nu}$. Then $\mathcal{B}$ gives $\mathsf{pk} = (g, g_1, g_2, f, h_0, h_1)$ to $\mathcal{A}$. Note that the helper keys are implicitly $\mathsf{mst}_\nu = g^{a^3} g^{a\cdot\gamma}$ and $\mathsf{mst}_{\overline{\nu}} = g_2^{\alpha_{\overline{\nu}}}$, where mst_ν is unknown to $\mathcal{B}$.

`Find Stage.` In this stage, $\mathcal{A}$ issues a series of queries, and $\mathcal{B}$ responds as follows:

- User secret key oracle $\mathcal{O}_{\mathrm{u}}(t)$: $\mathcal{B}$ outputs a random bit and **aborts** if $t = \ell$ or $(t = \ell - 1) \wedge (\overline{\nu} = \ell \mod 2)$. Otherwise, let $i = t \mod 2$, $j = (t-1) \mod 2$, and $\mathcal{B}$ responds to this query according to the following two cases:
  - Case $i = \nu$: Pick $r'_{i,t}, r_{j,t} \xleftarrow{\$} \mathbb{Z}_p^*$. If $\nu = \ell \mod 2$, generate usk_t for $\mathcal{A}$ as in Eqs. (5)-(7). Otherwise, generate $\mathsf{usk}_t = (g', (a_{j,t}, b_{j,t}), (a_{i,t}, b_{i,t}, c_{i,t}))$ for $\mathcal{A}$ as below:

$$
\begin{aligned}
a_{j,t} &= \mathsf{mst}_j \cdot \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r_{j,t}}, \quad b_{j,t} = g^{r_{j,t}}, \\
a_{i,t} &= g^{a\cdot\gamma} \cdot \left(g^{a^2}\right)^{-(\alpha'+\alpha_j + \frac{\omega_i - H(\ell)\theta}{H(t)-H(\ell-1)})} \cdot \left(g^{\alpha^4}\right)^{-\frac{H(\ell)}{H(t)-H(\ell-1)}} \cdot \left(g_1^{H(t)} h_i\right)^{r'_{i,t}}, \\
b_{i,t} &= \left(g^{a^2}\right)^{\frac{-1}{H(t)-H(\ell-1)}} g^{r'_{i,t}}, \; c_{i,t} = \left(g^{a^4}\right)^{\frac{-1}{H(t)-H(\ell-1)}} \left(g^{a^2}\right)^{r'_{i,t} - \frac{\theta}{H(t)-H(\ell-1)}} g^{\theta\cdot r'_{i,t}}.
\end{aligned}
$$

    Letting $r_{i,t} = \frac{-a^2}{H(t)-H(\ell-1)} + r'_{i,t}$, it is readily to see that the above user secret keys usk_t have the correct form as required.
  - Case $j = \nu$: Pick $r_{i,t}, r'_{j,t} \xleftarrow{\$} \mathbb{Z}_p^*$. If $\nu = \ell \mod 2$, generate usk_t for $\mathcal{A}$ as in Eqs. (8)-(10). Otherwise, generate $\mathsf{usk}_t = (g', (a_{j,t}, b_{j,t}), (a_{i,t}, b_{i,t}, c_{i,t}))$ for $\mathcal{A}$ as below:

$$
\begin{aligned}
a_{j,t} &= (g^a)^{\gamma - \theta - \frac{(\alpha'+\alpha_i)(H(t-1)-H(\ell-1))+\omega_j}{H(t)-H(\ell)}} \left(g^{a^2}\right)^{\frac{H(\ell-1)-H(t-1)}{H(t)-H(\ell)}} \left(g_1^{H(t-1)} f^{H(t)} h_j\right)^{r'_{j,t}}, \\
b_{j,t} &= (g^a)^{\frac{-1}{H(t)-H(\ell)}} g^{r'_{j,t}}, \\
a_{i,t} &= \mathsf{mst}_i \cdot \left(g_1^{H(t)} h_i\right)^{r_{i,t}}, \quad b_{i,t} = g^{r_{i,t}}, \; c_{i,t} = f^{r_{i,t}}.
\end{aligned}
$$

    Letting $r_{i,t} = \frac{-a}{H(t)-H(\ell)} + r'_{i,t}$, it is readily to see that the above user secret keys usk_t have the correct form as required.
- Helper key query $\mathcal{O}_{\mathrm{h}}(i)$ with $i \in \{0,1\}$: If $i = \nu$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ returns $\mathsf{mst}_{\overline{\nu}}$ to $\mathcal{A}$.

`Challenge.` Once $\mathcal{A}$ decides that guess stage is over, he outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathbb{G}_T$ on which it wishes to be challenged. If $t^* \neq \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ picks $\beta \xleftarrow{\$} \{0,1\}$, and defines $C_1^* = Z \cdot e\left(h, (g^a)^{\gamma} (g^{a^2} g^{\gamma})^{\alpha'+\alpha_{\overline{\nu}}}\right) \cdot m_\beta, C_2^* = h$. If $v = \ell \mod 2$, $\mathcal{B}$ defines $C_3^* = h^{\omega_{\overline{\nu}}}$ and

$C_4^* = h^{\omega_\nu}$; otherwise, $\mathcal{B}$ defines $C_3^* = h^{\omega_\nu}$ and $C_4^* = h^{\omega_{\overline{\nu}}}$. Finally, $\mathcal{B}$ gives the challenge ciphertext $\mathtt{CT}^* = (C_1^*, C_2^*, C_3^*, C_4^*)$ to $\mathcal{A}$. It is readily to see that, if $Z = e(g,h)^{a^3}$, the above challenge ciphertext $\mathtt{CT}^*$ has the correct form as required, and when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext $\mathtt{CT}^*$ is independent of β in the adversary's view.

`Guess stage.` $\mathcal{A}$ continues to adaptively issue additional queries as in find stage, and $\mathcal{B}$ also responds as in find stage.

`Output.` Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta' = \beta$, then $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

This completes the descriptions of the simulation. Similarly to the analysis in Lemma 3, we can see that $\mathcal{B}$'s advantage against the 2-DBDHE assumption is at least $\epsilon/4N$, and it is also clear that $\mathcal{B}$'s running time is polynomially bounded. Thus the proof of Lemma 5 is concluded. □

Next, we prove the strong-IND-KI-CPA-security for our Boneh-Boyen-Goh-based PKIE scheme under the the DBDH assumption. Note that it is easy to see that, if the 2-DBDHE assumption holds in groups $(\mathbb{G}, \mathbb{G}_T)$, the DBDH assumption certainly holds.

Lemma 6 *The Boneh-Boyen-Goh-based* PKIE *scheme presented in Figure 7 is* strong-IND-KI-CPA *secure, assuming the DBDH assumption holds in groups* $(\mathbb{G}, \mathbb{G}_T)$.

Proof. Given a polynomial time adversary $\mathcal{A}$ who can break the strong-IND-KI-CPA security of our Boneh-Boyen-Goh-based PKIE scheme with advantage ϵ, we indicate that we can construct a polynomial time algorithm $\mathcal{B}$ that can break the DBDH assumption in groups $(\mathbb{G}, \mathbb{G}_T)$ with advantage ϵ/N, where N denotes the total number of periods.

Given a DBDH instance $(g, g^a, g^b, g^c, Z) \in \mathbb{G}^4 \times \mathbb{G}_T$ with unknown $a, b, c \xleftarrow{\$} \mathbb{Z}_p^*$, $\mathcal{B}$'s goal is to decide whether $Z = e(g,g)^{abc}$. $\mathcal{B}$ acts as the challenger and provides the simulations for $\mathcal{A}$ in the experiment $\mathbf{Exp}^{\text{strong-IND-KI-CPA}}_{\mathcal{A},\mathtt{PKIE}}$ as follows:

`Init Stage.` $\mathcal{B}$ first chooses an index $\ell \xleftarrow{\$} \{1, \cdots, N\}$ as a guess for the target time period to be attacked by $\mathcal{A}$. Let $i^* = \ell \mod 2$ and $j^* = (\ell - 1) \mod 2$. Next, $\mathcal{B}$ picks $\alpha_0, \alpha_1, \mu_0, \mu_1 \xleftarrow{\$} \mathbb{Z}_p^*$, and defines $g_1 = g^a g^{\alpha_0+\alpha_1}, g_2 = g^b$, $h_{i^*} = g_1^{-H(\ell)} g^{\mu_{i^*}}$, $h_{j^*} = g_1^{-H(\ell-1)} f^{H(\ell)} g^{\mu_{j^*}}$, $\mathsf{mst}_1 = g_2^{\alpha_1}$ and $\mathsf{mst}_0 = g_2^{\alpha_0}$. Finally, $\mathcal{B}$ gives $\mathsf{pk} = (g, g_1, g_2, f, h_0, h_1)$ and $(\mathsf{mst}_1, \mathsf{mst}_0)$ to $\mathcal{A}$. Note that g' is implicitly set to be $g' = g^{ab}$ which is unknown to $\mathcal{B}$.

`Challenge.` $\mathcal{A}$ outputs a target period index t^* and two equal-length plaintexts $m_0, m_1 \in \mathbb{G}_T$ on which it wishes to be challenged. If $t^* \neq \ell$, $\mathcal{B}$ outputs a random bit and **aborts**. Otherwise, $\mathcal{B}$ picks $\beta \xleftarrow{\$} \{0,1\}$, and gives the challenge ciphertext $\mathtt{CT}^* = (C_1^*, C_2^*, C_3^*, C_4^*) = (Z \cdot e(g^c, g^b)^{\alpha_0+\alpha_1} \cdot m_\beta, g^c, (g^c)^{\mu_{j^*}}, (g^c)^{\mu_{i^*}})$ to $\mathcal{A}$.

Observe that, if $Z = e(g,g)^{abc}$ the above challenge ciphertext has the correct form as required, since

$$
\begin{aligned}
C_1^* &= Z \cdot e(g^c, g^b)^{\alpha_0+\alpha_1} \cdot m_\beta = e(g,g)^{abc} \cdot e(g^{\alpha_0+\alpha_1}, g^b)^c \cdot m_\beta \\
&= e(g^a, g^b)^c \cdot e(g^{\alpha_0+\alpha_1}, g^b)^c \cdot m_\beta = e(g^a g^{\alpha_0+\alpha_1}, g^b)^c \cdot m_\beta = e(g_1, g_2)^c \cdot m_\beta, \\
C_2^* &= g^c, \\
C_3^* &= (g^c)^{\mu_{j^*}} = (g^{\mu_{j^*}})^c = (g_1^{H(\ell-1)} f^{H(\ell)} g_1^{-H(\ell-1)} f^{-H(\ell)} g^{\mu_{j^*}})^c = (g_1^{H(t^*-1)} f^{H(t^*)} h_{j^*})^c, \\
C_4^* &= (g^c)^{\mu_{i^*}} = (g^{\mu_{i^*}})^c = (g_1^{H(\ell)} g_1^{-H(\ell)} g^{\mu_{i^*}})^c = (g_1^{H(t^*)} h_{i^*})^c.
\end{aligned}
$$

On the other hand, when Z is uniform and independent in $\mathbb{G}_T$, the challenge ciphertext $\mathtt{CT}^*$ is independent of β in the adversary's view.

Output. Finally, $\mathcal{A}$ outputs a guess $\beta' \in \{0,1\}$. If $\beta' = \beta$, then $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

Analysis: The simulations provided for $\mathcal{A}$ are perfect, if $\mathcal{B}$ correctly guesses $\ell = t^*$, whose probability is obviously at least $1/N$. Thus $\mathcal{B}$'s advantage against the DBDH assumption is at least ϵ/N. It is also clear that $\mathcal{B}$'s running time is polynomially bounded. Thus the proof of Lemma 6 is concluded. □

4.3 Extension: PKIE Scheme with n helpers

In this section, we discuss how to extend our concrete PKIE scheme to obtain a PKIE scheme with n helpers (n-PKIE for short). In such a scheme, these n helpers are *alternately* used to update the user secret key. For example, in period t, the helper with helper key $\mathsf{mst}_{t \mod n}$ is used to update the user secret key from usk_{t-1} to usk_t. While in the next period $t+1$, the helper key $\mathsf{mst}_{(t+1) \mod n}$ will be used to update user secret key from usk_t to usk_{t+1}. Similarly to PKIE systems, key-insulated security for an n-PKIE system captures the intuition that, even if up to $n-1$ helpers are broken into while a given period t is exposed, only one other period adjacent to t is exposed. Furthermore, the strong key-insulated security ensures that, even if the n helpers are simultaneously corrupted, all the periods are still secure. Formal definitions and security notions for n-PKIE is given in Appendix A.

Our PKIE scheme presented in Figure 5 can be naturally extended to an n-PKIE scheme as presented in Figure 8. We here explain the meanings of the subscripts used in the scheme. For the random numbers $r_{i,k}$ used in algorithms KeyGen and Δ-Gen, its subscript k corresponds to the time period index, and i corresponds to the index of the helper key mst_i. The subscripts in $(a_{i,t}, b_{i,t})$ and $(\hat{a}_{i,t}, \hat{b}_{i,t})$ have the similar meanings.

Since the n-PKIE Scheme in Figure 8 is a natural extension of the Boneh-Boyen-based PKIE scheme presented in Figure 5, its strong key-insulated security under chosen-plaintext attack can be proved under the DBDH assumption in a similar way as Theorem 4. Thus we here omit the detailed proofs.

<table>
<tr><td colspan="2">KeyGen(λ): Given a security parameter λ,
1. pick $g, g_2, h \xleftarrow{\$} \mathbb{G}$ and $\alpha', \alpha_0, \cdots, \alpha_{n-1} \xleftarrow{\$} \mathbb{Z}_p^*$, and set $g' = g_2^{\alpha'}, g_1 = g^{\alpha' + \sum_{i=0}^{n-1} \alpha_i}$,
2. set $\mathsf{mst}_i = g_2^{\alpha_i}$ for $i = 0$ to $n-1$,
3. pick a collision-resistant hash function $H : \mathbb{Z} \to \mathbb{Z}_p^*$,
4. for $k = 0$ to $n-1$,
compute $a_{0,k} = \big(\prod_{i=k}^{n-1} \mathsf{mst}_i\big) \cdot \big(g_1^{H(k)} h\big)^{\sum_{i=k}^{n-1} r_{i,k}}$ and $b_{0,k} = g^{\sum_{i=k}^{n-1} r_{i,k}}$, where $r_{i,k} \xleftarrow{\$} \mathbb{Z}_p^*$ for $i = k, \cdots, n-1$,
5. return $\mathsf{pk} = (g, g_1, g_2, h)$, $\mathsf{usk}_0 = (g', (a_{0,0}, b_{0,0}), \cdots, (a_{0,n-1}, b_{0,n-1}))$ and $(\mathsf{mst}_0, \cdots, \mathsf{mst}_{n-1})$.</td></tr>
<tr><td>Δ-Gen($t, \mathsf{mst}_{t \mod n}$): To generate the update key hsk_t with the matching helper key $\mathsf{mst}_{t \mod n}$,
1. let $i = t \mod n$,
2. for $k = t$ to $t+n-1$
pick $r_{i,k} \xleftarrow{\$} \mathbb{Z}_p^*$,
set $\hat{a}_{i,k} = \mathsf{mst}_i \cdot \big(g_1^{H(k)} h\big)^{r_{i,k}}$, $\hat{b}_{i,k} = g^{r_{i,k}}$,
3. return $\mathsf{hsk}_t = ((\hat{a}_{i,t}, \hat{b}_{i,t}), \cdots, (\hat{a}_{i,t+n-1}, \hat{b}_{i,t+n-1}))$.</td><td>Update($t, \mathsf{usk}_{t-1}, \mathsf{hsk}_t$): In period t, to update the user secret key from usk_{t-1} to usk_t,
1. let $i = t \mod n$ and $j = (t-1) \mod n$
2. parse usk_{t-1} as $(g', (a_{j,t-1}, b_{j,t-1}), \cdots, (a_{j,t+n-2}, b_{j,t+n-2}))$,
3. parse hsk_t as $((\hat{a}_{i,t}, \hat{b}_{i,t}), \cdots, (\hat{a}_{i,t+n-1}, \hat{b}_{i,t+n-1}))$,
4. for $k = t$ to $t+n-2$
set $a_{i,k} = a_{j,k} \cdot \hat{a}_{i,k}$, and $b_{i,k} = b_{j,k} \cdot \hat{b}_{i,k}$.
5. set $a_{i,t+n-1} = \hat{a}_{i,t+n-1}$ and $b_{i,t+n-1} = \hat{b}_{i,t+n-1}$
6. return $\mathsf{usk}_t = (g', (a_{i,t}, b_{i,t}), \cdots, (a_{i,t+n-1}, b_{i,t+n-1}))$.</td></tr>
<tr><td>Enc(pk, t, m): In period t, to encrypt a message m under public key pk,
1. pick $s \xleftarrow{\$} \mathbb{Z}_p^*$, and compute $C_1 = e(g_1, g_2)^s \cdot m$, $C_2 = g^s$, $C_3 = (g_1^{H(t)} h)^s$,
2. return $\mathtt{CT} = (C_1, C_2, C_3)$.</td><td>Dec($\mathsf{usk}_t, \mathtt{CT}$): To decrypt a ciphertext CT with the matching user secret key usk_t,
1. parse CT as (C_1, C_2, C_3),
2. parse usk_t as $(g', (a_{i,t}, b_{i,t}), \cdots, (a_{i,t+n-1}, b_{i,t+n-1}))$,
3. return $m \leftarrow \dfrac{C_1 \cdot e(C_3, b_{i,t})}{e\,(C_2, g' \cdot a_{i,t})}$.</td></tr>
</table>

Figure 8: n-PKIE Scheme Based on Boneh-Boyen IBE

4.4 Comparisons

In this section, we compare the efficiency of LQY scheme (which is the currently known best scheme in the standard model) and our proposed PKIE schemes in terms of communication overhead and computational cost. In Table 1, $|\mathbb{G}|$ and $|\mathbb{G}_T|$ denote the bit-length of an element in group $\mathbb{G}$ and $\mathbb{G}_T$ respectively, t_r, t_m, t_T and t_p denote the computational cost of one regular exponentiation in $\mathbb{G}$, one multi-exponentiation [39] in $\mathbb{G}$, one regular exponentiation in $\mathbb{G}_T$ and one pairing in $(\mathbb{G}, \mathbb{G}_T)$ respectively, and "without-RO" denotes that the security is proved without random oracles. We note that t_m is approximately equal to $1.2t_r$ due to the Pippenger algorithm [39].

	LQY PKIE [36]	BB-based PKIE	BBG-based PKIE	LQY n-PKIE [36]	BB-based n-PKIE
public key	$6\|\mathbb{G}\|$	$4\|\mathbb{G}\|$	$6\|\mathbb{G}\|$	$(2n+2)\|\mathbb{G}\|$	$4\|\mathbb{G}\|$
user secret key	$3\|\mathbb{G}\|$	$5\|\mathbb{G}\|$	$6\|\mathbb{G}\|$	$(n+1)\|\mathbb{G}\|$	$(2n+1)\|\mathbb{G}\|$
ciphertext	$3\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$2\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$3\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$(n+1)\|\mathbb{G}\|+1\|\mathbb{G}_T\|$	$2\|\mathbb{G}\|+1\|\mathbb{G}_T\|$
Δ-Gen	$2t_m+1t_r$	$2t_m+2t_r$	$1t_m+2t_r$	$2t_m+1t_r$	nt_m+nt_r
Enc	$2t_m+1t_r+1t_T$	$1t_m+1t_r+1t_T$	$2t_m+1t_r+1t_T$	$nt_m+1t_r+1t_T$	$1t_m+1t_r+1t_T$
Dec	$3t_p$	$2t_p$	$3t_p$	$(n+1)t_p$	$2t_p$
without RO?	✓	✓	✓	✓	✓
strong security?	×	✓	✓	×	✓

Table 1: Comparisons between our concrete schemes and LQY schemes [36]

As indicated in Table 1, our proposed schemes are secure in the strengthened security model, while LQY scheme is only secure in the weak model. Our Boneh-Boyen-based scheme is superior to other schemes in many aspects (except for size of user secret key and computational cost for Δ-Gen). In terms of computational cost for Δ-Gen, our Boneh-Boyen-Goh-based PKIE scheme is superior to other schemes, and it is suitable for environments where helpers are computationally weak. LQY scheme is superior to others in terms of size of user secret key. In summary, we have three different PKIE schemes with different advantages, and one can choose an appropriate one from them according to each situation.

As to the extended schemes with n helpers, the advantage of our Boneh-Boyen-based n-PKIE scheme over (the n-PKIE version of) LQY scheme becomes more obvious. The public key size, ciphertext size, encryption cost and decryption cost in the n-PKIE version of LQY scheme grows linearly with the number n of helpers, while ours are independent of the number of helpers. Honestly, we admit that our scheme still has the following limitations: the computation cost in algorithm Δ-Gen is linear with the number of helpers, and its user secret key size is about twice of LQY scheme. It would be an interesting open problem to construct a PKIE scheme with constant public key size, ciphertext size, encryption cost and decryption cost, and yet with shorter user secret key and lower computational cost for algorithm Δ-Gen.

Acknowledgements

The authors would like to thank Yevgeniy Dodis as a part of this work was inspired by discussion with him. The authors would also like to thank Benoît Libert for his valuable comments. This work is supported by the National Science Foundation of China under Grant No. 60903178, and it also supported by the Fundamental Research Funds for the Central Universities.

A Parallel Key-Insulated Encryption with n Helpers

Similarly to standard PKIE, a parallel key-insulated encryption scheme with n helpers consists of five algorithms, i.e., (KeyGen, Δ-Gen, Update, Enc, Dec), where the last three algorithms are the same as those in standard PKIE, and the first two algorithms are specified as follows:

- $\mathsf{KeyGen}(\lambda)$: The *key generation* algorithm, on input a security parameter λ, outputs a public key pk, an initial user secret key usk_0 and n helper keys $(\mathsf{mst}_0 \cdots, \mathsf{mst}_{n-1})$. We write $(\mathsf{pk}, \mathsf{usk}_0, (\mathsf{mst}_0 \cdots, \mathsf{mst}_{n-1})) \leftarrow \mathsf{KeyGen}(\lambda)$.
- $\Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \mod n})$: The *helper key-update* algorithm, on input a period index t and the corresponding helper key $\mathsf{mst}_{t \mod n}$, it outputs an update key hsk_t for period t. We write $\mathsf{hsk}_t \leftarrow \Delta\text{-}\mathsf{Gen}(t, \mathsf{mst}_{t \mod n})$.

The key-insulated security for an n-PKIE scheme should capture the intuition that, even if up to $n-1$ helpers are broken into while a given period t is exposed, only one other period adjacent to t is exposed. We refer to this security as IND-nKI-ATK where $\mathsf{ATK} \in \{\mathsf{CCA}, \mathsf{CPA}\}$. For an adversary $\mathcal{A}$, we consider the following experiment:

$$\mathbf{Exp}^{\mathsf{IND\text{-}nKI\text{-}ATK}}_{\mathcal{A},\text{n-PKIE}}(\lambda): \big[(\mathsf{pk}, \mathsf{usk}_0, (\mathsf{mst}_0, \cdots, \mathsf{mst}_{n-1})) \leftarrow \mathsf{KeyGen}(\lambda);\ (m_0, m_1, t^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{u}}(\cdot),\mathcal{O}_{\mathrm{h}}(\cdot),\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{find}}(\mathsf{pk});$$
$$\beta \xleftarrow{\$} \{0,1\}, \mathtt{CT}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, t^*, m_\beta);\ \beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{u}}(\cdot),\mathcal{O}_{\mathrm{h}}(\cdot),\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{guess}}(\mathsf{pk}, \mathtt{CT}^*);\ \text{return 1 if } \beta = \beta' \text{ or 0 otherwise}\big],$$

where $\mathcal{O}_{\mathrm{u}}(\cdot)$ and $\mathcal{O}_{\mathrm{d}}$ are the same as in experiment $\mathbf{Exp}^{\mathsf{IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\mathsf{PKIE}}(\lambda)$, and $\mathcal{O}_{\mathrm{h}}(\cdot)$ is a helper key oracle which for given an index $i \in \{0, 1, \cdots, n-1\}$ returns the helper key hsk_i. It is mandated that $|m_0| = |m_1|$, and the following requirements should be simultaneously satisfied:

(1). $\mathcal{A}$ cannot issue the user secret key query $\mathcal{O}_{\mathrm{u}}(t^*)$;
(2). $\mathcal{A}$ cannot issue both queries $\mathcal{O}_{\mathrm{u}}(t^*-1)$ and $\mathcal{O}_{\mathrm{h}}(t^* \mod n)$;
(3). $\mathcal{A}$ cannot issue both queries $\mathcal{O}_{\mathrm{u}}(t^*+1)$ and $\mathcal{O}_{\mathrm{h}}((t^*+1) \mod n)$;
(4). $\mathcal{A}$ cannot corrupt all of the n helpers;
(5). If $\mathsf{ATK} = \mathsf{CCA}$, $\mathcal{A}$ cannot issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*, \mathtt{CT}^*)$.

We define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathsf{IND\text{-}KI\text{-}ATK}}_{\mathcal{A},\text{n-PKIE}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{IND\text{-}nKI\text{-}ATK}}_{\mathcal{A},\mathsf{PKIE}}(\lambda) = 1] - \frac{1}{2}\right|$.

Definition 9 *We say that an* n-PKIE *scheme is* IND-nKI-CCA *(resp.* IND-nKI-CPA*) secure, if there exists no PPT adversary $\mathcal{A}$ who has non-negligible advantage* $\mathrm{Adv}^{\mathsf{IND\text{-}nKI\text{-}CCA}}_{\mathcal{A},\mathsf{PKIE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{IND\text{-}nKI\text{-}CPA}}_{\mathcal{A},\mathsf{PKIE}}(\lambda)$*).*

The strong key-insulated security for an n-PKIE scheme should further ensure that, breaking into all of the n helpers does not help the adversary as long as he does not additionally obtain any user secret keys for any periods. To define this security notion, we first define the strong-IND-nKI-ATK-security, where $\mathsf{ATK} \in \{\mathsf{CCA}, \mathsf{CPA}\}$. For an adversary $\mathcal{A}$, we consider the following experiment:

$$\mathbf{Exp}^{\mathsf{strong\text{-}IND\text{-}nKI\text{-}ATK}}_{\mathcal{A},\text{n-PKIE}}(\lambda): \big[(\mathsf{pk}, \mathsf{usk}_0, (\mathsf{mst}_0, \cdots, \mathsf{mst}_{n-1})) \leftarrow \mathsf{KeyGen}(\lambda);$$
$$(m_0, m_1, t^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{find}}(\mathsf{pk}, \mathsf{mst}_0, \cdots, \mathsf{mst}_{n-1}); \beta \xleftarrow{\$} \{0,1\};\ \mathtt{CT}^* \leftarrow \mathsf{Enc}(\mathsf{pk}, t^*, m_\beta);$$
$$\beta' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)}_{\mathtt{guess}}(\mathsf{pk}, \mathsf{mst}_0, \cdots, \mathsf{mst}_{n-1}, \mathtt{CT}^*);\ \text{return 1 if } \beta = \beta' \text{ or 0 otherwise}]$$

where $\mathcal{O}_{\mathrm{d}}(\cdot,\cdot)$ is the same as in experiment $\mathbf{Exp}^{\mathsf{IND\text{-}nKI\text{-}ATK}}_{\mathcal{A},\text{n-PKIE}}(\lambda)$. It is mandated that $|m_0| = |m_1|$, and if $\mathsf{ATK} = \mathsf{CCA}$ then $\mathcal{A}$ cannot issue the decryption query $\mathcal{O}_{\mathrm{d}}(t^*, \mathtt{CT}^*)$. We define $\mathcal{A}$'s advantage as $\mathrm{Adv}^{\mathsf{strong\text{-}IND\text{-}nKI\text{-}ATK}}_{\mathcal{A},\text{n-PKIE}}(\lambda) = \left|\Pr[\mathbf{Exp}^{\mathsf{strong\text{-}IND\text{-}nKI\text{-}ATK}}_{\mathcal{A},\text{n-PKIE}}(\lambda) = 1] - \frac{1}{2}\right|$.

Definition 10 *We say that an* n-PKIE *scheme is* strong-IND-nKI-CCA *(resp.* strong-IND-nKI-CPA*) secure, if there exists no PPT adversary $\mathcal{A}$ who has non-negligible advantage* $\mathrm{Adv}^{\mathsf{strong\text{-}IND\text{-}nKI\text{-}CCA}}_{\mathcal{A},\text{n-PKIE}}(\lambda)$ *(resp.* $\mathrm{Adv}^{\mathsf{strong\text{-}IND\text{-}nKI\text{-}CPA}}_{\mathcal{A},\text{n-PKIE}}(\lambda)$*).*

Definition 11 *We say that an* n-PKIE *scheme is strongly key-insulated secure under chosen-ciphertext attack (resp. chosen-plaintext attack), if it is both* IND-nKI-CCA *secure (resp.* IND-nKI-CPA *secure) and* strong-IND-nKI-CCA *secure (resp.* strong-IND-nKI-CPA *secure).*