Parallel Key-Insulated Public Key Encryption

Goichiro Hanaoka[1], Yumiko Hanaoka[2], and Hideki Imai[1,3]

[1] Research Center for Information Security,
National Institute of Advanced Industrial Science and Technology
1102 Akihabara Daibiru, 1-18-13 Sotokanda, Chiyoda-ku, Tokyo 101-0021, Japan.
hanaoka-goichiro@aist.go.jp
[2] NTT DoCoMo, Inc.
3-5 Hikarino-oka, Yokosuka 239-8536, Japan.
hanaoka@nttdocomo.co.jp
[3] Institute of Industrial Science, University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo 153-8505, Japan.
imai@iis.u-tokyo.ac.jp

Abstract. Security is constantly been infringed by inadvertent loss of secret keys, and as a solution, Dodis, Katz, Xu, and Yung [11], in Eurocrypt 2002, proposed a new paradigm called key-insulated security which provides tolerance against key exposures. Their scheme introduces a "helper key" which is used to periodically update the decryption key. The most attractive part of this scheme is that even if a decryption key of a time period is exposed, the security of the rest of the periods are unaffected. But how does this helper key managed? Can it be done efficiently? As, to alleviate the damage caused by key exposures, decryption key has to be updated at very short intervals, although frequent updating will, in contrary, increase the risk of helper key exposure. In this paper, we propose *parallel key-insulated public key encryption* in which two distinct helper keys *alternately* update a decryption key. The helper key of one system is independent from the other. Not only does it decrease the chance of helper key exposures, it also allows frequent updating of the decryption key, and over all, increases the security of the system.

1 Introduction

Background. The problem of key exposure is an important issue in practice, regardless. No matter how strong the encryption scheme is, if the key is exposed, we lose the security. Leaving it just under users' care and responsibility is too high of a risk to take: Loss of important documents and personal information that results from careless handling by humans happens nothing out of the ordinary, and the damage exerted, can be immeasurable. Classic approach was to try to earn the time before system collapse, although not solving the problem fundamentally, secret keys leaked eventually. Dodis, Katz, Xu and Yung looked at this problem from a different prospective: Their idea was to minimize the damage instead of just trying to gain time. Their proposed scheme was called, key-insulated public-key encryption (KIPE) [11]. In their KIPE, a helper key is introduced and is stored in the helper device which is kept isolated from the network except for times it is used to update the decryption key. Encryption in KIPE is carried out using a fixed public key and time (e.g. date), and so, the need to announce new public key to others after each key updating (like what the certificate authority does in PKI) can be omitted. Regarding its security, security for all time periods except for time period exposed, both forward and backward security, are guaranteed.

Now, to increase the system tolerance against key exposures for KIPE system, the first thing that comes into mind is to update the decryption key at short timing (i.e. frequent intervals), however, this will, in turn, increase the frequency of helper device connection to the network and increase the risk of helper key exposure. This is due to that the KIPE assumes helper key exposures to less likely occur than decryption key exposures: Helper key and decryption key are managed independently and helper device can be stored in a physically much safer place as it is used only at key updates. Frequent updating of the decryption key will in turn put the helper key in a higher risk of exposure (as it will be connected to the network more often). So, a trade-off between decryption key and helper key exposures exists. For deeper understanding, let's consider the next example: Suppose you are a busy office worker who wishes to increase system tolerance by frequently updating the key. You think that updating twice each day is manageable. You decide to update the key once at home (at approx. midnight) and once at the office (at approx. noon). Since you leave the helper device at home, now, you will need to remind yourself not to forget to bring the device to work each day, or otherwise, make a copy of the device and leave one copy at your office for convenience. In either case, security of the decryption key is increased but the risk of helper key exposure is also increased (doubled). So, as we can see, unless the KIPE model is changed, it is impossible to increase the security of both decryption key and helper key simultaneously.

Our Results. In this paper, we propose *parallel key-insulated public key encryption* (PKIPE). Our PKIPE allows frequent updating of the decryption key, and at the same time, reduces the risk of helper key exposure. PKIPE differs from the original KIPE in that two distinct helper devices are introduced and each device is alternately used to update a single decryption key (so, you don't have to carry your helper device to-and-from work and office each day).

Initialization in PKIPE involves providing two auxiliary *helpers* H_1 and H_2 with master helper keys mst_1 and mst_2, respectively, and the user's terminal with a *stage* 0 *user secret key* usk_0. Similarly to the original KIPE, user's public encryption key pk is treated like that of an ordinary encryption scheme with regard to certification, but its lifetime is divided into stages $i = 1, 2, ..., N(= 2n)$ with encryption in stage i performed as a function of pk, i and the plaintext, and decryption in stage i performed by using a *stage* i *user secret key* usk_i obtained by the following key-update process performed at the beginning of stage i:

- If $i = 2k-1$ for $k \in \{1, 2, ..., n\}$, H_1 sends to the user's terminal over a secure channel, a *stage* i *helper key* hsk_i computed as a function of mst_1 and i,
- If $i = 2k$ for $k \in \{1, 2, ..., n\}$, similarly to the above, H_2 sends hsk_i computed as a function of mst_2 and i,

the user computes usk_i as a function of usk_{i-1} and hsk_i, and erases usk_{i-1}. Like the original KIPE, our PKIPE also address random access key update [11] in which the user computes an arbitrary stage user secret key (that could also be a past key).

The security intentions are:

1. If none of the helpers is compromised, similar to the original KIPE, exposure of any of user secret keys does not compromise the security of the non-exposed stages, and

2. even if one of H_1 and H_2 is compromised in addition to the exposure of any of user secret keys, it still does not compromise the security of the non-exposed stages except for the ones whose corresponding user secret keys can be trivially determined from the exposed keys.

For case **2.**, consider a situation where an adversary obtains mst_1, usk_{i_0} and usk_{i_1} such that i_0 and i_1 are even and odd, respectively. Obviously, stages i_0 and i_1 are compromised. The security of stage $i_0 + 1$ may also be compromised since usk_{i_0+1} is easily computable from usk_{i_0} and mst_1. Similarly, security of stage $i_1 - 1$, too, may be compromised. (Notice that we address random access key update and so we can recover past keys). On the other hand, for example, the security of stage $i_1 + 1$ is not compromised as usk_{i_1+1} is computed as a function of usk_{i_1} and mst_2, and not mst_1. So, in this case, security of all stages except for i_0, $i_0 + 1$, i_1 and $i_1 - 1$ remain secure. Furthermore, if only one of H_1 or H_2 is compromised but none of the user secret key is exposed, then all stages remain secure. In other words, even for the case when one of helper keys, mst_1 and mst_2 is exposed, the security of our PKIPE is guaranteed to maintain at least the security level of the original KIPE.

Similar to the original KIPE, we can further address the case when all of the helper keys are exposed:

3. Even if both helpers H_1 and H_2 are compromised, security of all stages remain secure as long as user secret key (of even one stage) is not compromised in addition to the helper keys.

Our proposed schemes are proven to be semantically secure in the random oracle model.

Related Works. Followed by the earlier proposal made by Dodis, Katz, Xu and Yung [11], Dodis, Franklin, Katz, Miyaji and Yung proposed an *intrusion-resilient public key encryption* (IRPKE) [13] which strengthened the forward security of KIPE. The security of IRPKE has enhanced, only, it became less convenient as it did not allow random access key update. There were proposal of signature schemes as well with the same intention to provide tolerance against key exposures: Key-insulated signature [12] and intrusion-resilient signature [17]. On the other hand, as an encryption scheme that allows key update, there is, the KIPE and also, *forward secure public key encryption* (FSPKE). FSPKE was introduced by Anderson [1] and the first efficient construction was proposed by Canetti, Halevi and Katz [10]. Dodis, Franklin, Katz, Miyaji and Yung showed that by using FSPKE with a homomorphic property, a generic IRPKE can be constructed [14]. Not to mention, many variations of *forward secure signatures* have also been introduced, e.g. [8, 2].

Identity-based encryption (IBE) [18, 5, 9] works as a crucial building block in the construction of KIPE. Bellare and Palacio showed in [7] that a KIPE (OT-KIPE)[1] which allows unlimited number of key updating is equivalent to an IBE, and so, constructing a provably secure OT-KIPE in the standard model with [3], [4] or [19] can be done also. In this paper, we show that [5] is used as a basic building block to construct PKIPE.

2 Definitions

First, we give the model of PKIPE and the security notion. We follow by showing the characteristics of bilinear maps and a related computational assumption.

2.1 Model: Parallel Key-Insulated Public Key Encryption

A PKIPE scheme $\mathcal{E}$ consists of five efficient algorithms (**KeyGen, Δ-Gen, Update, Encrypt, Decrypt**).

KeyGen: Takes a security parameter k and returns mst_1, mst_2, usk_0 and pk. Public key pk includes a description of finite message space $\mathcal{M}$, and description of finite ciphertext space $\mathcal{C}$.

Δ-Gen: Takes as inputs, mst_j and i, and returns stage i helper key hsk_i if $j = i \bmod 2$, or $\perp$ otherwise.

Update: Takes as inputs, usk_{i-1}, hsk_i and i, and returns stage i user secret key usk_i.

Encrypt: Takes as inputs, pk, i and $M \in \mathcal{M}$, and returns ciphertext $C \in \mathcal{C}$.

Decrypt: Takes as inputs, pk, usk_i and $C \in \mathcal{C}$, and returns $M \in \mathcal{M}$ or $\perp$.

These algorithms satisfy $\forall i \in \{1, ..., N\}$, $\forall M \in \mathcal{M}$, $\mathbf{Decrypt}(pk, usk_i, C) = M$ where $C = \mathbf{Encrypt}(pk, i, M)$.

2.2 Security Notion

Here, we define the notion of semantic security for PKIPE. This is based on the security definition in the original KIPE [11, 7]. It should be noticed that the definition in [7] looks simpler than in [11] but they are essentially the same.

We say that a PKIPE scheme $\mathcal{E}$ is *semantically secure against an adaptive chosen ciphertext attack under an adaptive chosen key exposure attack* (IND-KE-CCA) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage against the challenger in the following IND-KE-CCA game:

Setup: The challenger takes a security parameter k and runs the **KeyGen** algorithm. He gives pk to $\mathcal{A}$ and keeps usk_0, mst_1 and mst_2 to himself.

Phase 1: $\mathcal{A}$ issues queries $q_1, \cdots, q_m$ where each of the queries q_i is one of:

- Exposure query $\langle j, \texttt{class} \rangle$: If `class` = "user", the challenger responds by running the algorithms **Δ-Gen** and **Update** to generate usk_j and sends it to $\mathcal{A}$. If `class` = "helper", the challenger sends mst_j to $\mathcal{A}$.

[1] key-insulated public key encryption with optimum threshold

– Decryption query $\langle j, C\rangle$: The challenger responds by running the algorithms **Δ-Gen** and **Update** to generate usk_j. He then runs **Decrypt** to decrypt the ciphertext C using usk_j and sends the result to $\mathcal{A}$.

These queries may be asked adaptively, that is, each query q_i may depend on the replies to $q_1, \cdots, q_{i-1}$.

Challenge: Once $\mathcal{A}$ decides that Phase 1 is over, she outputs two equal length plaintexts $M_0, M_1 \in \mathcal{M}$ and $j^* \in \{1, 2, ..., N\}$ on which she wishes to be challenged. The challenger picks a random bit $\beta \in \{0, 1\}$ and sets $C^* =$ **Encrypt**(pk, j^*, M_β). The challenger sends C^* as the challenge to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ issues additional queries $q_{m+1}, \cdots, q_{max}$ where each of the queries is one of:

– Exposure query $\langle j, \mathtt{class}\rangle$: Challenger responds as in Phase 1.
– Decryption query $\langle j, C\rangle$: Challenger responds as in Phase 1.

These queries may be asked adaptively as in Phase 1.

Guess: Finally, $\mathcal{A}$ outputs $\beta' \in \{0, 1\}$. She wins the game if $\beta' = \beta$ and

1. $\langle j^*, C^*\rangle$ does not appear in Decryption queries,
2. $\langle j^*,$ "user"$\rangle$ does not appear in Exposure queries,
3. both $\langle j^*-1,$ "user"$\rangle$ and $\langle 2-(j^* \bmod 2),$ "helper"$\rangle$ do not simultaneously appear in Exposure queries,
4. both $\langle j^*+1,$ "user"$\rangle$ and $\langle (j^* \bmod 2)+1,$ "helper"$\rangle$ do not simultaneously appear in Exposure queries,
5. both $\langle 1,$ "helper"$\rangle$ and $\langle 2,$ "helper"$\rangle$ do not simultaneously appear in Exposure queries.

We refer to such an adversary $\mathcal{A}$ as an IND-KE-CCA adversary. We define $\mathcal{A}$'s advantage in attacking the scheme $\mathcal{E}$ as $Adv_{\mathcal{E},\mathcal{A}} = \Pr[\beta' = \beta] - 1/2$. The provability is over the random bits used by the challenger and $\mathcal{A}$. As usual, we can define chosen plaintext security similarly to the game above except that the adversary is not allowed to issue any Decryption queries. We call this adversary IND-KE-CPA *adversary.*

Definition 1 We say that a PKIPE system $\mathcal{E}$ is (t, ϵ)-*adaptive chosen ciphertext secure under adaptive chosen key exposure attacks* if for any t-time IND-KE-CCA adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is IND-KE-CCA *secure.* Also, we say that $\mathcal{E}$ is (t, ϵ)-*adaptive chosen plaintext secure under adaptive chosen key exposure attacks* if for any t-time IND-KE-CPA adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is IND-KE-CPA *secure.*

IND-KE-CCA is already a strong security notion, but its security can be enhanced further to cover the compromise of both the helper keys. Concretely, as a constraint on the above adversary's Exposure query, we can modify 5. so that:

5′. $\langle 1,$ "helper"$\rangle$, $\langle 2,$ "helper"$\rangle$, and $\langle j,$ "user"$\rangle$ do not simultaneously appear in Exposure queries for any $j \in \{1, 2, ..., N\}$.

Such modification allows $\mathcal{A}$ to obtain both mst_1 and mst_2 if $\mathcal{A}$ doesn't ask any of user secret keys. Let this adversary be a *strong* IND-KE-CCA adversary. Similarly, we can define *strong* IND-KE-CPA *adversary*, and here as well, she is not allowed to issue any Decryption queries.

Definition 2 We say that a PKIPE system $\mathcal{E}$ is (t, ϵ)*-adaptive chosen ciphertext secure under strongly adaptive chosen key exposure attacks* if for any t-time strong $\mathsf{IND\text{-}KE\text{-}CCA}$ adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is *strongly* $\mathsf{IND\text{-}KE\text{-}CCA}$ *secure.* Also, we say that $\mathcal{E}$ is (t, ϵ)*-adaptive chosen plaintext secure under strongly adaptive chosen key exposure attacks* if for any t-time strong $\mathsf{IND\text{-}KE\text{-}CPA}$ adversary $\mathcal{A}$, we have $Adv_{\mathcal{E},\mathcal{A}} < \epsilon$. As shorthand, we say that $\mathcal{E}$ is *strongly* $\mathsf{IND\text{-}KE\text{-}CPA}$ *secure.*

A Remark. In the discussion we had so far, it may seem like we may have overlooked the exposure of stage i helper key, but actually, we haven't. It is obvious that if hsk_i can be computed from usk_{i-1} and usk_i for any stage i, then exposure of hsk_i can be emulated by using the responses to the Exposure queries. So, the security definition so far given is sufficient as it is even against exposure of stage i helper keys for any i, if we assume that such property holds. As a matter of fact, all of our constructions satisfy this property.

2.3 Bilinear Maps and the CBDH Assumption

Throughout this paper, we let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order q, and g be a generator of $\mathbb{G}_1$. A *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ satisfies the following properties: (i) For all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$. (ii) $e(g, g) \neq 1$. (iii) There is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}_1$. The *Computational Bilinear Diffie-Hellman (CBDH) problem* [5] in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ is as follows: given a tuple $(g, g^a, g^b, g^c) \in (\mathbb{G}_1)^4$ as input, output $e(g, g)^{abc} \in \mathbb{G}_2$. An algorithm $\mathcal{A}_{cbdh}$ solves CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ with the probability ϵ_{cbdh} if $\Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc}] \geq \epsilon_{cbdh}$, where the probability is over the random choice of generator $g \in \mathbb{G}_1 \backslash \{1\}$, and $a, b, c \in \mathbb{Z}_q$ and random coins consumed by $\mathcal{A}_{cbdh}$.

Definition 3 We say that the $(t_{cbdh}, \epsilon_{cbdh})$*-CBDH assumption* holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ if no t_{cbdh}-time algorithm has advantage of at least ϵ_{cbdh} in solving the CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$.

3 Chosen Plaintext Secure Construction

In this section, we propose our PKIPE schemes and prove its security under CBDH assumption in the random oracle model. Intuitively, picture two independent Boneh-Franklin IBEs (BF-IBE) [5, 6] integrated to one another and the master key of one BF-IBE is free to leak. Applying a straightforward 2-out-of-2 threshold key generation of BF-IBE [5] is not the correct answer since then the master keys of both BF-IBEs will be required to update a decryption key. Instead, in our PKIPE schemes, master keys of the two independent BF-IBEs are *alternately* used to update a single key (so, only one master key is used at a time). Furthermore, interestingly, decryption key size, ciphertext size and computational cost for decryption in our PKIPE remain unchanged (and public key size and encryption cost is increased but only slightly for one element in $\mathbb{G}_1$ and one pairing computation, respectively) as in the original BF-IBE. In our schemes, we let $N = O(\mathsf{poly}(k))$.

3.1 Construction

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two groups of order q of size k, and g be a generator of $\mathbb{G}_1$. Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. Let G, H be cryptographic hash functions $G : \mathbb{G}_2 \to \{0,1\}^n$ for some n, $H : \{0,1\}^* \to \mathbb{G}_1$, respectively. The message space is $\mathcal{M} = \{0,1\}^n$. The PKIPE1 scheme consists of the following algorithms:

PKIPE1: IND-KE-CPA CONSTRUCTION

KeyGen: Given a security parameter k, **KeyGen** algorithm:

1. generates $\mathbb{G}_1$, $\mathbb{G}_2$, g and e.
2. picks $s_1, s_2 \in \mathbb{Z}_q^*$ uniformly at random, and sets $h_1 = g^{s_1}$ and $h_2 = g^{s_2}$,
3. chooses cryptographic hash functions G and H,
4. computes $u_{-1} = H(-1)$ and $u_0 = H(0)$,
5. computes $d_{-1} = u_{-1}^{s_1}$ and $d_0 = u_0^{s_2}$,
6. outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H \rangle$, $mst_1 = s_1$, $mst_2 = s_2$ and $usk_0 = d_{-1} \cdot d_0$.

Δ-Gen: For given mst_j and $i \in \{1, 2, ..., N\}$, **Δ-Gen** algorithm:

1. outputs $\perp$ if $i \not\equiv j \bmod 2$,
2. computes $u_{i-2} = H(i-2)$ and $u_i = H(i)$,
3. computes $d_{i-2} = u_{i-2}^{s_j}$ and $d_i = u_i^{s_j}$,
4. outputs $hsk_i = d_{i-2}^{-1} \cdot d_i$.

Update: For given usk_{i-1}, hsk_i and i, **Update** algorithm:

1. computes $usk_i = usk_{i-1} \cdot hsk_i$,
2. deletes usk_{i-1} and hsk_i,
3. outputs usk_i.

Encrypt: For given pk, i and a message $M \in \{0,1\}^n$, **Encrypt** algorithm:

1. chooses random $r \in \mathbb{Z}_q^*$,
2. computes $u_{i-1} = H(i-1)$ and $u_i = H(i)$,
3. if $i = 0 \bmod 2$, computes $W = (e(h_1, u_{i-1}) \cdot e(h_2, u_i))^r$,
4. if $i = 1 \bmod 2$, computes $W = (e(h_1, u_i) \cdot e(h_2, u_{i-1}))^r$,
5. sets $C = \langle i,\ g^r,\ G(W) \oplus M \rangle$,
6. outputs C as a ciphertext.

Decrypt: For given pk, usk_i and $C = \langle i, c_0, c_1 \rangle$, **Decrypt** algorithm:

1. computes $W' = e(c_0, usk_i)$,
2. computes $M' = c_1 \oplus G(W')$,
3. outputs M' as a plaintext.

3.2 Security

Now, we prove that PKIPE1 is IND-KE-CPA under the CBDH assumption. For readers who are already familiar with KIPE and/or IBE, here we give an overview of the proof. PKIPE1 is based on [5, 6], so, a proof technique similar to [5, 6] can be applied. However, there are still some technical hurdles to overcome due to the peculiar key-updating mechanism using two different helper keys. Namely, embedding the given CBDH instance into the responses to the adversary's queries cannot be straightforwardly carried out since the keys are mutually dependent on one another, and that the simulation fails if inconsistency of the responses is noticed by the adversary. For example, suppose that the simulator embeds the given instance into $usk_\alpha (= d_{\alpha-1}d_\alpha)$ for some stage α. Here, the simulator does not know the value of usk_α but has to respond to any Exposure queries (except for usk_α) including $usk_{\alpha-1} (= d_{\alpha-2}d_{\alpha-1})$ and $usk_{\alpha+1} (= d_\alpha d_{\alpha+1})$. We notice that both factors of usk_α, i.e. $d_{\alpha-1}$ and d_α appear in $usk_{\alpha-1}$ or $usk_{\alpha+1}$, and so, responding to $usk_{\alpha-1}$ and $usk_{\alpha+1}$ without knowing usk_α is not easy.

Theorem 1 *Suppose $(t_{cbdh}, \epsilon_{cbdh})$-CBDH assumption holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and hash functions G and H are random oracles. Then,* PKIPE1 *is $(t_{pkipe}, \epsilon_{pkipe})$-*IND-KE-CPA *secure as long as $\epsilon_{pkipe} \le \frac{3q_G N}{2}\epsilon_{cbdh}$ and $t_{pkipe} \le t_{cbdh} + \Theta(\tau(2q_H + 3q_E))$, where* IND-KE-CPA *adversary $\mathcal{A}_{pkipe}$ issues at most q_H H-queries and q_E Exposure queries. Here, τ is the maximum time for computing an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$, and pairing e.*

Proof. We show that we can construct an algorithm $\mathcal{A}_{cbdh}$ that can solve the CBDH problem in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ by using an adversary $\mathcal{A}_{pkipe}$ that breaks IND-KE-CPA security of our scheme. The algorithm $\mathcal{A}_{cbdh}$ is given an instance $\langle g, g^a, g^b, g^c\rangle$ in $\mathbb{G}_1$ from the challenger and tries to output $e(g,g)^{abc}$ using $\mathcal{A}_{pkipe}$. Let $g_1 = g^a, g_2 = g^b, g_3 = g^c$. The algorithm $\mathcal{A}_{cbdh}$ works by interacting with $\mathcal{A}_{pkipe}$ in an IND-KE-CPA game as follows:

Before starting the simulation, $\mathcal{A}_{cbdh}$ flips a coin $\mathcal{COIN} \in \{0,1\}$ such that we have $\Pr[\mathcal{COIN} = 0] = \delta$ for some δ which we will determine later. If $\mathcal{COIN} = 0$, $\mathcal{A}_{cbdh}$ simulates the responses to $\mathcal{A}_{pkipe}$'s queries expecting that $\mathcal{A}_{pkipe}$ will never submit $\langle j, \text{“helper”}\rangle$ as Exposure query for any j. If $\mathcal{COIN} = 1$, $\mathcal{A}_{cbdh}$ carries out the simulation expecting that $\mathcal{A}_{pkipe}$ will submit $\langle j, \text{“helper”}\rangle$ for some j.

If $\mathcal{COIN} = 0$, $\mathcal{A}_{cbdh}$ responses to $\mathcal{A}_{pkipe}$'s queries will be as follows:

Setup: $\mathcal{A}_{cbdh}$ picks a random $s \in \mathbb{Z}_q^*$. Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{pkipe}$ the system parameter $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H\rangle$, where $h_1 = g_1$ and $h_2 = g_1^s$, and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below.

G-queries: $\mathcal{A}_{pkipe}$ issues up to q_G queries to the random oracle G. To respond to these queries algorithm, $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle W, x\rangle$ as explained below. We call this list G_{list}. The list is initially empty. When $\mathcal{A}_{pkipe}$ gives $\mathcal{A}_{cbdh}$ a query W to the oracle G, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query W already appears on the G_{list} in a tuple $\langle W, x\rangle$, then outputs $G(W) = x$.
2. $\mathcal{A}_{cbdh}$ chooses a random $x \in \{0,1\}^n$.
3. $\mathcal{A}_{cbdh}$ adds the tuple $\langle W, x\rangle$ to the G_{list} and outputs $G(W) = x$.

H-queries: $\mathcal{A}_{cbdh}$ picks a random $\alpha \in \{1, ..., N\}$ in advance. $\mathcal{A}_{pkipe}$ issues up to q_H queries to the random oracle H. To respond to these queries algorithm, $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle i, u_i, r_i\rangle$ as explained below. We call the list H_{list}. The list is initially empty. When $\mathcal{A}_{pkipe}$ gives $\mathcal{A}_{cbdh}$ a query i to the oracle H, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query i already appears on the H_{list} in a tuple $\langle i, u_i, r_i\rangle$, then outputs $H(i) = u_i$.

2. If $i = \alpha$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_\alpha = 0$.
3. If $i < \alpha$, $\mathcal{A}_{cbdh}$ chooses a random $r_i \in \mathbb{Z}_q^*$ and sets $u_i = g^{r_i}$.
4. If $i > \alpha$, $\mathcal{A}_{cbdh}$ chooses a random $r_i \in \mathbb{Z}_q^*$ and sets $u_i = g_2^z \cdot g^{r_i}$, where
   - $z = 1$ if $i = \alpha \bmod 2$,
   - $z = -s$ if $i = 1 \bmod 2$ and $\alpha = 0 \bmod 2$,
   - $z = -s^{-1}$ if $i = 0 \bmod 2$ and $\alpha = 1 \bmod 2$, where s^{-1} is the inverse of $s \bmod q$,
5. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, u_i, r_i \rangle$ to the H_{list} and outputs $H(i) = u_i$.

Challenge: Once algorithm $\mathcal{A}_{pkipe}$ decides that Phase 1 is over, it outputs a target stage i^* and two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{A}_{cbdh}$ responds as follows:
1. $\mathcal{A}_{cbdh}$ sets $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as $c_0^* = g_3$ and $c_1^* = \mu$, where $\mu \in_R \{0,1\}^n$.
2. $\mathcal{A}_{cbdh}$ gives $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as the challenge ciphertext to $\mathcal{A}_{pkipe}$.

Exposure queries: $\mathcal{A}_{pkipe}$ issues up to q_E Exposure queries. When $\mathcal{A}_{pkipe}$ gives a query $\langle i, \texttt{class} \rangle$, $\mathcal{A}_{cbdh}$ responds as follows:
1. If `class` = "helper" or $i = \alpha$, $\mathcal{A}_{cbdh}$ aborts the simulation.
2. $\mathcal{A}_{cbdh}$ runs the algorithm for responding to H-queries to obtain $\langle i, u_i, r_i \rangle$ and $\langle i-1, u_{i-1}, r_{i-1} \rangle$.
3. $\mathcal{A}_{cbdh}$ sets $usk_i = h_1^{r_{i-1}} \cdot h_2^{r_i}$ if $i = 0 \bmod 2$, or $usk_i = h_1^{r_i} \cdot h_2^{r_{i-1}}$ otherwise, and outputs usk_i to $\mathcal{A}_{pkipe}$. Observe that usk_i is the user secret key corresponding to the stage i. Especially, when $i > \alpha$,

$$\begin{aligned}
u_{i-1}^{\log_g h_1} \cdot u_i^{\log_g h_2} &= (g_2^{-s} \cdot g^{r_{i-1}})^a \cdot (g_2 \cdot g^{r_i})^{s \cdot a} = h_1^{r_{i-1}} h_2^{r_i} \\
&\qquad\qquad (\text{if } i = 0 \bmod 2,\ \alpha = 0 \bmod 2) \\
&= (g_2 \cdot g^{r_{i-1}})^a \cdot (g_2^{-s^{-1}} \cdot g^{r_i})^{s \cdot a} = h_1^{r_{i-1}} h_2^{r_i} \\
&\qquad\qquad (\text{if } i = 0 \bmod 2,\ \alpha = 1 \bmod 2) \\
u_{i-1}^{\log_g h_2} \cdot u_i^{\log_g h_1} &= (g_2 \cdot g^{r_{i-1}})^{s \cdot a} \cdot (g_2^{-s} \cdot g^{r_i})^a = h_2^{r_{i-1}} h_1^{r_i} \\
&\qquad\qquad (\text{if } i = 1 \bmod 2,\ \alpha = 0 \bmod 2) \\
&= (g_2^{-s^{-1}} \cdot g^{r_{i-1}})^{s \cdot a} \cdot (g_2 \cdot g^{r_i})^a = h_2^{r_{i-1}} h_1^{r_i} \\
&\qquad\qquad (\text{if } i = 1 \bmod 2,\ \alpha = 1 \bmod 2)
\end{aligned}$$

Guess: When $\mathcal{A}_{pkipe}$ decides that Phase 2 is over, $\mathcal{A}_{pkipe}$ outputs its guess bit $\beta' \in \{0,1\}$. At the same time, algorithm $\mathcal{A}_{cbdh}$ terminates the simulation. Then, $\mathcal{A}_{cbdh}$ picks a tuple $\langle W, x \rangle$ uniformly at random from the G_{list}, and computes $T = (\frac{W}{e(g_1,g_3)^{r_{\alpha-1}}})^{s^{-1}}$ if $\alpha = 0 \bmod 2$, or $T = (\frac{W}{e(g_1,g_3)^{s \cdot r_{\alpha-1}}})$ if $\alpha = 1 \bmod 2$. Finally, $\mathcal{A}_{cbdh}$ outputs T.

Claim 1 *If $i^* = \alpha$ and $\mathcal{A}_{cbdh}$ does not abort, then $\mathcal{A}_{pkipe}$'s view is identical to its view in the real attack until $\mathcal{A}_{pkipe}$ submits W^* as a G-query, where $W^* = e(g_1, g_3)^{r_{\alpha-1}} \cdot e(g,g)^{s \cdot abc}$ if $\alpha = 0 \bmod 2$, or $W^* = e(g_1, g_3)^{s \cdot r_{\alpha-1}} \cdot e(g,g)^{abc}$ if $\alpha = 1 \bmod 2$.*

Proof. It is obvious that the responses to G and H are as in the real attack. Interestingly, the responses to Exposure queries are perfect if $\mathcal{A}_{cbdh}$ does not abort. Finally, we show that the response to Challenge is indistinguishable from the real attack until $\mathcal{A}_{pkipe}$ submits W^*. Let the response to Challenge be $C^* = \langle \alpha, c_0^*, c_1^* \rangle$. Then, c_0^* is uniformly distributed in $\mathbb{G}_1$ due to random $\log_g g_3 (= c)$, and therefore are as in the real attack. Also, since $c_1^* = M_\beta \oplus G(W^*)$, it is information-theoretically impossible to obtain any information on M_β unless $\mathcal{A}_{pkipe}$ asks $G(W^*)$. □

Next, let us define by E_1, an event assigned to be true if and only if $i^* = \alpha$. Similarly, let us define by E_2, an event assigned to be true if and only if a G-query coincides with W^*, and by E_{msk}, an event assigned to be true if and only if an Exposure query coincides with $\langle i, \text{“helper”} \rangle$ for any $i \in \{1, 2\}$.

Claim 2 *We have that* $\Pr[\beta' = \beta | E_1, \neg E_{msk}] \geq \Pr[\beta' = \beta | \neg E_{msk}]$.

Proof. It is clear that $\sum_{i \in \{1,\ldots,N\}} \Pr[\beta' = \beta | i^* = i, \neg E_{msk}] \Pr[i^* = i | \neg E_{msk}] = \Pr[\beta' = \beta | \neg E_{msk}]$. Since α is uniformly chosen from $\{1, ...N\}$ at random, we have $\Pr[\beta' = \beta | i^* = \alpha, \neg E_{msk}]] \Pr[i^* = \alpha | \neg E_{msk}] \geq \frac{1}{N} \Pr[\beta' = \beta | \neg E_{msk}]$. Therefore, we have $\Pr[\beta' = \beta | E_1, \neg E_{msk}] \geq \Pr[\beta' = \beta | \neg E_{msk}]$, which proves the claim. □

Claim 3 *We have that* $\Pr[\beta' = \beta | E_1, \neg E_2, \neg E_{msk}] = 1/2$.

Proof. Let C^* be $\langle \alpha, c_0^*, c_1^* \rangle$. Since $c_1^* = M_\beta \oplus G(W^*)$, it is impossible to obtain any information on M_β without asking W^* as a G-query. □

Claim 4 *We have that* $\Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc} | \mathcal{COIN} = 0] \geq \frac{1}{q_G N} \cdot \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}]$.

Proof. If $i^* = \alpha$, then $e(g, g)^{abc}$ can easily be calculated from W^*, and W^* appears in G_{list} with probability $\Pr[E_2]$. We have $\Pr[E_2] \geq \Pr[E_2 | E_1, \neg E_{msk}] \cdot \Pr[E_1 | \neg E_{msk}] \cdot \Pr[\neg E_{msk}]$ and $\Pr[E_1 | \neg E_{msk}] = 1/N$. Hence, by choosing a tuple from G_{list} uniformly at random, $\mathcal{A}_{cbdh}$ can correctly output $e(g, g)^{abc}$ with probability of at least $1/q_G \cdot 1/N \cdot \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}]$. □

Finally, we calculate $p_0 := \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc} | \mathcal{COIN} = 0]$. Letting $\gamma := \Pr[\beta' = \beta | E_{msk}] - 1/2$, from Claims 1 and 2, we have

$$\begin{aligned}
\Pr[\beta' = \beta] - \frac{1}{2} &= \Pr[\beta' = \beta | \neg E_{msk}] \Pr[\neg E_{msk}] + \Pr[\beta' = \beta | E_{msk}] \Pr[E_{msk}] - \frac{1}{2} \\
&= \Pr[\beta' = \beta | \neg E_{msk}](1 - \Pr[E_{msk}]) + (\frac{1}{2} + \gamma) \Pr[E_{msk}] - \frac{1}{2} \\
&\leq \Pr[\beta' = \beta | E_1, \neg E_{msk}](1 - \Pr[E_{msk}]) + (\frac{1}{2} + \gamma) \Pr[E_{msk}] - \frac{1}{2}.
\end{aligned}$$

From $\Pr[\beta' = \beta | E_1, \neg E_{msk}] = \Pr[\beta' = \beta | E_1, E_2, \neg E_{msk}] \cdot \Pr[E_2 | E_1, \neg E_{msk}] + \Pr[\beta' = \beta | E_1, \neg E_2, \neg E_{msk}] \cdot \Pr[\neg E_2 | E_1, \neg E_{msk}]$ and Claim 3, we have

$$\begin{aligned}
\Pr[\beta' = \beta] - \frac{1}{2} &\leq (\Pr[E_2 | E_1, \neg E_{msk}] + \frac{1}{2}(1 - \Pr[E_2 | E_1, \neg E_{msk}]))(1 - \Pr[E_{msk}]) \\
&\quad + (\frac{1}{2} + \gamma) \Pr[E_{msk}] - \frac{1}{2} \\
&= \frac{1}{2} \Pr[E_2 | E_1, \neg E_{msk}] \Pr[\neg E_{msk}] + \gamma \Pr[E_{msk}].
\end{aligned}$$

From Claim 4, we have $p_0 \geq \frac{2}{q_G N}(\epsilon_{pkipe} - \gamma \Pr[E_{msk}])$.

Next, we discuss for the $\mathcal{COIN} = 1$ case. If $\mathcal{COIN} = 1$, $\mathcal{A}_{cbdh}$ responses to $\mathcal{A}_{pkipe}$'s queries as follows:

Setup: $\mathcal{A}_{cbdh}$ picks random $s \in \mathbb{Z}_q^*$ and $\mathbf{b} \in \{1, 2\}$. Let $\bar{\mathbf{b}}$ be 1 (resp. 2) if $\mathbf{b} = 2$ (resp. 1). Also, $\mathcal{A}_{cbdh}$ gives $\mathcal{A}_{pkipe}$ the system parameter $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H \rangle$, where $h_{\mathbf{b}} = g_1$ and $h_{\bar{\mathbf{b}}} = g^s$, and random oracles G, H are controlled by $\mathcal{A}_{cbdh}$ as described below.

G-queries: $\mathcal{A}_{pkipe}$ issues up to q_G queries to the random oracle G. To respond to these queries algorithm $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle W, x \rangle$ as explained below. We call this list G_{list}. The list is initially empty. When $\mathcal{A}_{pkipe}$ gives $\mathcal{A}_{cbdh}$ a query W to the oracle G, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query W already appears on the G_{list} in a tuple $\langle W, x \rangle$, then outputs $G(W) = x$.
2. $\mathcal{A}_{cbdh}$ chooses a random $x \in \{0, 1\}^n$.
3. $\mathcal{A}_{cbdh}$ adds the tuple $\langle W, x \rangle$ to the G_{list} and outputs $G(W) = x$.

H-queries: $\mathcal{A}_{cbdh}$ picks a random $\alpha \in \{1, ..., N\}$ in advance. $\mathcal{A}_{pkipe}$ issues up to q_H queries to the random oracle H. To respond to these queries algorithm $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle i, u_i, r_i \rangle$ as explained below. We call the list H_{list}. The list is initially empty. When $\mathcal{A}_{pkipe}$ gives $\mathcal{A}_{cbdh}$ a query i to the oracle H, $\mathcal{A}_{cbdh}$ responds as follows:

1. If the query i already appears on the H_{list} in a tuple $\langle i, u_i, r_i \rangle$, then outputs $H(i) = u_i$.
2. If $i = \alpha - 1$ and $\alpha \equiv \bar{\mathbf{b}} \bmod 2$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_i = 0$.
3. If $i = \alpha$ and $\alpha \equiv \mathbf{b} \bmod 2$, $\mathcal{A}_{cbdh}$ sets $u_i = g_2$ and $r_i = 0$.
4. Else, $\mathcal{A}_{cbdh}$ chooses a random $r_i \in \mathbb{Z}_q^*$ and sets $u_i = g^{r_i}$.
5. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, u_i, r_i \rangle$ to the H_{list} and outputs $H(i) = u_i$.

Challenge: Once algorithm $\mathcal{A}_{pkipe}$ decides that Phase 1 is over, it outputs a target stage i^* and two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{A}_{cbdh}$ responds as follows:

1. $\mathcal{A}_{cbdh}$ sets $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as $c_0^* = g_3$ and $c_1^* = \mu$, where $\mu \in_R \{0, 1\}^n$.
2. $\mathcal{A}_{cbdh}$ gives $C^* = \langle i^*, c_0^*, c_1^* \rangle$ as the challenge ciphertext to $\mathcal{A}_{pkipe}$.

Exposure queries: $\mathcal{A}_{pkipe}$ issues up to q_E Exposure queries. When $\mathcal{A}_{pkipe}$ gives a query $\langle i, \texttt{class} \rangle$, $\mathcal{A}_{cbdh}$ responds as follows:

1. If $i = \mathbf{b}$ and $\texttt{class} =$ "helper", $\mathcal{A}_{cbdh}$ aborts the simulation.
2. If $i = \bar{\mathbf{b}}$ and $\texttt{class} =$ "helper", $\mathcal{A}_{cbdh}$ returns s to $\mathcal{A}_{pkipe}$.
3. If $i = \alpha$ and $\texttt{class} =$ "user", $\mathcal{A}_{cbdh}$ aborts the simulation.
4. If $i = \alpha - 1$, $\texttt{class} =$ "user" and $\alpha \equiv \bar{\mathbf{b}} \bmod 2$, $\mathcal{A}_{cbdh}$ aborts the simulation.
5. If $i = \alpha + 1$, $\texttt{class} =$ "user" and $\alpha \equiv \mathbf{b} \bmod 2$, $\mathcal{A}_{cbdh}$ aborts the simulation.
6. Else[2], $\mathcal{A}_{cbdh}$ runs the algorithm for responding to H-queries to obtain $\langle i, u_i, r_i \rangle$ and $\langle i-1, u_{i-1}, r_{i-1} \rangle$, and sets $usk_i = h_1^{r_{i-1}} \cdot h_2^{r_i}$ if $i = 0 \bmod 2$, or $usk_i = h_1^{r_i} \cdot h_2^{r_{i-1}}$ otherwise. $\mathcal{A}_{cbdh}$ outputs usk_i to $\mathcal{A}_{pkipe}$.

[2] Notice that in this case, $\texttt{class}$ is always "user".

Guess: When $\mathcal{A}_{pkipe}$ decides that Phase 2 is over, $\mathcal{A}_{pkipe}$ outputs the guess bit $\beta' \in \{0,1\}$. At the same time, algorithm $\mathcal{A}_{cbdh}$ terminates the simulation. Then, $\mathcal{A}_{cbdh}$ picks a tuple $\langle W, x\rangle$ uniformly at random from the G_{list}, and computes $T = W \cdot e(g, g_3)^{-s \cdot r_\alpha}$ if $\alpha \equiv \bar{\mathbf{b}} \bmod 2$, or $T = W \cdot e(g, g_3)^{-s \cdot r_{\alpha-1}}$ if $\alpha \equiv \mathbf{b} \bmod 2$. Finally, $\mathcal{A}_{cbdh}$ outputs T.

Claim 5 *If $i^* = \alpha$, $\mathcal{A}_{pkipe}$ submits $\langle \bar{\mathbf{b}}$, "helper"$\rangle$ as an Extraction query, and $\mathcal{A}_{cbdh}$ does not abort, then $\mathcal{A}_{pkipe}$'s view is identical to its view in the real attack until $\mathcal{A}_{pkipe}$ submits W^* as a G-query, where $W^* = e(g, g_3)^{s \cdot r_\alpha} \cdot e(g, g)^{abc}$ if $\alpha \equiv \bar{\mathbf{b}} \bmod 2$, or $W^* = e(g, g_3)^{s \cdot r_{\alpha-1}} \cdot e(g, g)^{abc}$ if $\alpha \equiv \mathbf{b} \bmod 2$.*

Next, let us define by E_3, an event assigned to be true if and only if $i^* = \alpha$. Similarly, let us define by E_4, an event assigned to be true if and only if a G-query coincides with W^*, by E_5, an event assigned to be true if and only if an Exposure query coincides with $\langle \mathbf{b}$, "helper"$\rangle$, and by E_{msk}, an event assigned to be true if and only if an Exposure query coincides with $\langle i$, "helper"$\rangle$ for any $i \in \{1, 2\}$. Notice that E_{msk} is identical to that in the case of $\mathcal{COIN} = 0$.

Claim 6 *We have that* $\Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}] \geq \Pr[\beta' = \beta | E_{msk}]$.

Claim 7 *We have that* $\Pr[\beta' = \beta | E_3, \neg E_4, \neg E_5, E_{msk}] = 1/2$.

Proofs of Claims 5, 6 and 7 are given in the full version of this paper.

Claim 8 *We have that* $\Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc} | \mathcal{COIN} = 1] \geq \frac{1}{2q_G N} \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}]$.

Proof. If $i^* = \alpha$, then $e(g, g)^{abc}$ can easily be calculated from W^*, and W^* appears in G_{list} with probability $\Pr[E_4]$. We have $\Pr[E_4] \geq \Pr[E_4 | E_3, \neg E_5, E_{msk}] \cdot \Pr[E_3 | \neg E_5, E_{msk}] \cdot \Pr[\neg E_5, E_{msk}]$. Furthermore, we have $\Pr[E_3 | \neg E_5, E_{msk}] = 1/N$, and $\Pr[\neg E_5, E_{msk}] = 1/2 \cdot \Pr[E_{msk}]$. Hence, by choosing a tuple from G_{list} uniformly at random, $\mathcal{A}_{cbdh}$ can correctly output $e(g, g)^{abc}$ with probability of at least $1/q_G \cdot 1/N \cdot 1/2 \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] \Pr[E_{msk}]$. □

Finally, we calculate $p_1 := \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc} | \mathcal{COIN} = 1]$. Letting $\eta := \Pr[\beta' = \beta | \neg E_{msk}] - 1/2$, from Claims 5 and 6, we have

$$\begin{aligned}
\Pr[\beta' = \beta] - \frac{1}{2} &= \Pr[\beta' = \beta | \neg E_{msk}] \Pr[\neg E_{msk}] + \Pr[\beta' = \beta | E_{msk}] \Pr[E_{msk}] - \frac{1}{2} \\
&= (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] + \Pr[\beta' = \beta | E_{msk}](1 - \Pr[\neg E_{msk}]) - \frac{1}{2} \\
&\leq (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] \\
&\quad + \Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}](1 - \Pr[\neg E_{msk}]) - \frac{1}{2}.
\end{aligned}$$

Since we have $\Pr[\beta' = \beta | E_3, \neg E_5, E_{msk}] = \Pr[\beta' = \beta | E_3, E_4, \neg E_5, E_{msk}] \cdot \Pr[E_4 | E_3, \neg E_5, E_{msk}] + \Pr[\beta' = \beta | E_3, \neg E_4, \neg E_5, E_{msk}] \cdot \Pr[\neg E_4 | E_3, \neg E_5, E_{msk}]$, from Claim 7, we have

$$\begin{aligned}\Pr[\beta' = \beta] - \frac{1}{2} \le & (\frac{1}{2} + \eta) \Pr[\neg E_{msk}] \\ & +(\Pr[E_4|E_3, \neg E_5, E_{msk}] + \frac{1}{2}(1 - \Pr[E_4|E_3, \neg E_5, E_{msk}])) \\ & \cdot(1 - \Pr[\neg E_{msk}]) - \frac{1}{2} \\ = & \frac{1}{2} \Pr[E_4|E_3, \neg E_5, E_{msk}] \Pr[E_{msk}] + \eta \Pr[\neg E_{msk}].\end{aligned}$$

From Claim 8, we have $p_1 \ge \frac{1}{q_G N}(\epsilon_{pkipe} - \eta \Pr[\neg E_{msk}])$.

Claim 9 *We have that* $\epsilon_{pkipe} \ge \gamma \Pr[E_{msk}] + \eta \Pr[\neg E_{msk}]$.

Proof. By the definitions of γ and η, we have $\gamma + 1/2 = \Pr[\beta' = \beta|E_{msk}]$ and $\eta + 1/2 = \Pr[\beta' = \beta|\neg E_{msk}]$, and consequently, $\epsilon_{pkipe} + \frac{1}{2} \ge \Pr[\beta' = \beta] = (\gamma + \frac{1}{2}) \Pr[E_{msk}] + (\eta + \frac{1}{2}) \Pr[\neg E_{msk}]$. Hence, we have $\epsilon_{pkipe} \ge \gamma \Pr[E_{msk}] + \eta \Pr[\neg E_{msk}]$, which proves the claim. □

Now, we calculate $\epsilon_{cbdh}(= \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g, g)^{abc}])$. From Claim 9, we have

$$\begin{aligned}\epsilon_{cbdh} &= \delta \cdot p_0 + (1 - \delta) \cdot p_1 \\ &\ge \delta(\frac{2}{q_G N}(\epsilon_{pkipe} - \gamma \Pr[E_{msk}])) + (1 - \delta)(\frac{1}{q_G N}(\epsilon_{pkipe} - \eta \Pr[\neg E_{msk}])) \\ &\ge \delta(\frac{2}{q_G N}(\epsilon_{pkipe} - \gamma \Pr[E_{msk}])) + (1 - \delta)(\frac{1}{q_G N}\gamma \Pr[E_{msk}]) \\ &\ge \frac{1}{q_G N}(2\delta\epsilon_{pkipe} + (1 - 3\delta)\gamma \Pr[E_{msk}])\end{aligned}$$

By letting $\delta = 1/3$, we finally have $\epsilon_{cbdh} \ge \frac{2}{3q_G N}\epsilon_{pkipe}$.

From the above discussions, we can see that the claimed bound of the running-time of $\mathcal{A}_{cbdh}$ holds. This completes the proof of the theorem. □

3.3 Strongly IND-KE-CPA Scheme

We can build a construction of a strongly IND-KE-CPA scheme PKIPE2 by only slightly modifying PKIPE1. The PKIPE2 consists of the following algorithms:

PKIPE2: STRONGLY IND-KE-CPA CONSTRUCTION

KeyGen: Given a security parameter k, **KeyGen** algorithm does the same as that of PKIPE1 except that it:

2. picks random, $s_1, s_2, s_3 \in \mathbb{Z}_q^*$, and sets $h_1 = g^{s_1 s_3}$ and $h_2 = g^{s_2 s_3}$,
6. outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, G, H\rangle$, $mst_1 = s_1$, $mst_2 = s_2$ and $usk_0 = \langle d_{-1}^{s_3} \cdot d_0^{s_3}, s_3\rangle$.

***Δ*-Gen:** Same as in PKIPE1.

Update: For given $usk_{i-1} = \langle usk'_{i-1}, s_3 \rangle$, hsk_i and i, **Update** algorithm:
1. computes $usk'_i = usk'_{i-1} \cdot hsk_i^{s_3}$,
2. deletes usk'_{i-1} and hsk_i,
3. outputs $usk_i = \langle usk'_i, s_3 \rangle$.

Encrypt: Same as in PKIPE1.

Decrypt: For given $usk_i = \langle usk'_i, s_3 \rangle$ and $C = \langle i, c_0, c_1 \rangle$, **Decrypt** algorithm does the same as that of PKIPE1 except that it:
1. computes $W' = e(c_0, usk'_i)$.

The security proof of PKIPE2 can be done similarly to PKIPE1. Here we briefly explain why both master keys, mst_1 and mst_2, can be exposed and still guarantee security. Since plaintext M is perfectly hidden by $G(e(g^r, usk'_i))$, it is necessary to compute $e(g^r, usk'_i)$ for compromising semantic security of PKIPE2. However, this is almost as difficult as the CBDH problem without knowing s_3 even if the adversary knows both mst_1 and mst_2. Hence, PKIPE2 is more secure than PKIPE1 against exposure of master helper keys.

4 Chosen Ciphertext Secure Construction

In this section, we construct chosen ciphertext secure PKIPE schemes by extending PKIPE1 and PKIPE2 with Fujisaki-Okamoto padding [15, 16]. It should be noticed that the proofs of security of our schemes cannot be straightforwardly done since the model of PKIPE significantly differs from the standard public key encryption.

4.1 Construction

Let F, G, H be cryptographic hash functions $F : \{1, ..., N\} \times \{0,1\}^n \times \{0,1\}^\ell \rightarrow \mathbb{Z}_q^*$, $G : \mathbb{G}_2 \rightarrow \{0,1\}^{n+\ell}$ for some n and ℓ, and $H : \{0,1\}^* \rightarrow \mathbb{G}_1$, respectively. The message space is $\mathcal{M} = \{0,1\}^n$. Except for the ones that are mentioned, the notions are the same as in PKIPE1. The PKIPE3 consists of the following algorithms:

PKIPE3: IND-KE-CCA CONSTRUCTION

KeyGen: Given a security parameter k, **KeyGen** algorithm does the same as that of PKIPE1 exceptthat it:

3. chooses cryptographic hash functions F, G and H,
6. outputs $pk = \langle q, \mathbb{G}_1, \mathbb{G}_2, e, n, g, h_1, h_2, F, G, H \rangle$, $mst_1 = s_1$, $mst_2 = s_2$ and $usk_0 = d_{-1} \cdot d_0$.

***Δ*-Gen:** Same as in PKIPE1.

Update: Same as in PKIPE1.

Encrypt: For given pk, i and a message $M \in \{0,1\}^n$, **Encrypt** algorithm:
1. chooses random $R \in \{0,1\}^\ell$,
2. computes $\sigma = F(i, M, R)$,
3. computes $u_{i-1} = H(i-1)$ and $u_i = H(i)$,
4. if $i = 0 \bmod 2$, computes $W = (e(h_1, u_{i-1}) \cdot e(h_2, u_i))^\sigma$,

5. if $i = 1 \bmod 2$, computes $W = (e(h_1, u_i) \cdot e(h_2, u_{i-1}))^\sigma$,
6. sets $C = \langle i,\ g^\sigma,\ G(W) \oplus (M||R)\rangle$,
7. outputs C as a ciphertext.

Decrypt: For given pk, usk_i and $C = \langle i, c_0, c_1\rangle$, **Decrypt** algorithm:
1. outputs $\perp$ if $C \notin \mathbb{Z}_N \times \mathbb{G}_1 \times \{0,1\}^{n+\ell}$,
2. computes $W' = e(c_0, usk_i)$,
3. computes $(M'||R') = c_1 \oplus G(W')$,
4. outputs M' as a plaintext if $c_0 = g^{\sigma'}$, or $\perp$ otherwise, where $\sigma' = F(i, M', R')$.

4.2 Security

Now, we prove that PKIPE3 is IND-KE-CCA under the CBDH assumption.

Theorem 2 *Suppose $(t_{cbdh}, \epsilon_{cbdh})$-CBDH assumption holds in $\langle \mathbb{G}_1, \mathbb{G}_2, e\rangle$ and hash functions G and H are random oracles. Then,* PKIPE3 *is $(t_{pkipe}, \epsilon_{pkipe})$-*IND-KE-CCA *secure as long as $\epsilon_{pkipe} \le \frac{3q_G N}{2}\epsilon_{cbdh} + \frac{2q_F}{2^\ell} + \frac{2q_D}{q}$ and $t_{pkipe} \le t_{cbdh} + \Theta(\tau(5q_F + 2q_H + 3q_E + 5q_D))$, where* IND-KE-CPA *adversary $\mathcal{A}_{pkipe}$ issues at most q_F F-queries, q_H H-queries, q_D Decryption queries and q_E Exposure queries. Here, τ is the maximum time for computing an exponentiation in $\mathbb{G}_1, \mathbb{G}_2$, and pairing e.*

Proof. The proof of theorem is almost identical to Theorem 1 except that here, $\mathcal{A}_{cbdh}$ has to simulate responses to Decryption queries as well. For either the case for $\mathcal{COIN} = 0$ and 1, if $i \neq \alpha$, then it will be easy for $\mathcal{A}_{cbdh}$ to calculate usk_i on his own, so the decryption will be easily done as well. Therefore, we only need to consider the case for $i = \alpha$. Concretely, $\mathcal{A}_{cbdh}$'s responses to $\mathcal{A}_{pkipe}$'s queries can be simulated as follows:

F-queries: $\mathcal{A}_{pkipe}$ picks a random $R^* \in \{0,1\}^\ell$ in advance. $\mathcal{A}_{pkipe}$ issues up to q_F queries to the random oracle F. To respond to these queries, algorithm $\mathcal{A}_{cbdh}$ forms a list of tuples $\langle i, M, R, \sigma\rangle$ as explained below. We call this list F_{list}. The list is initially empty. When $\mathcal{A}_{pkipe}$ gives $\mathcal{A}_{cbdh}$ a query (i, M, R) to the oracle F, $\mathcal{A}_{cbdh}$ responds as follows:
1. If $R = R^*$, $\mathcal{A}_{cbdh}$ aborts the simulation.
2. If the query (i, M, R) already appears on the F_{list} in a tuple $\langle i, M, R, \sigma\rangle$, then outputs $F(i, M, R) = \sigma$.
3. $\mathcal{A}_{cbdh}$ chooses a random $\sigma \in \mathbb{Z}_q^*$.
4. $\mathcal{A}_{cbdh}$ adds the tuple $\langle i, M, R, \sigma\rangle$ to the F_{list} and outputs $F(i, M, R) = \sigma$.

Challenge: Once algorithm $\mathcal{A}_{pkipe}$ decides that Phase 1 is over, it outputs a target stage i^* and two messages M_0, M_1 on which it wishes to be challenged. Algorithm $\mathcal{A}_{cbdh}$ responds as follows:
1. $\mathcal{A}_{cbdh}$ sets $C^* = \langle i^*, c_0^*, c_1^*\rangle$ as $c_0^* = g_3$ and $c_1^* = \mu$, where $\mu \in_R \{0,1\}^{n+\ell}$.
2. $\mathcal{A}_{cbdh}$ gives $C^* = \langle i^*, c_0^*, c_1^*\rangle$ as the challenge ciphertext to $\mathcal{A}_{pkipe}$.

Decryption queries: $\mathcal{A}_{pkipe}$ issues up to q_D Decryption queries. When $\mathcal{A}_{pkipe}$ gives a query $C = \langle i, c_0, c_1\rangle$, $\mathcal{A}_{cbdh}$ responds as follows:

1. If $i \neq \alpha$, $\mathcal{A}_{cbdh}$ runs the algorithm to respond to Exposure queries to obtain usk_i, decrypts C, and outputs the decryption result to $\mathcal{A}_{pkipe}$.
2. If $i = \alpha$, $\mathcal{A}_{cbdh}$ searches for a tuple $\langle \alpha, M, R, \sigma \rangle$ from F_{list} such that

$$\begin{aligned} c_0 &= g^{\sigma}, \\ c_1 &= G((e(h_1, u_{\alpha-1}) \cdot e(h_2, u_{\alpha}))^{\sigma}) \oplus (M||R) \qquad \text{if } \alpha = 0 \bmod 2, \\ &= G((e(h_1, u_{\alpha}) \cdot e(h_2, u_{\alpha-1}))^{\sigma}) \oplus (M||R) \qquad \text{if } \alpha = 1 \bmod 2. \end{aligned}$$

3. If there exists such a tuple, $\mathcal{A}_{cbdh}$ outputs M to $\mathcal{A}_{pkipe}$. Otherwise, $\mathcal{A}_{cbdh}$ outputs $\perp$.

Responses to G-queries, H-queries and Exposure queries can be simulated similarly to the proof in Theorem 1.

Next, let us define by $F\text{-}Fail$ an event assigned to be true if and only if there exists a F-query $\langle i, M, R \rangle$ such that $R = R^*$. Similarly, let us define by $D\text{-}Fail$ an event assigned to be true if and only if $\mathcal{A}_{cbdh}$ returns $\perp$ for a Decryption query which should not be rejected.

Using this, and following the proof of Theorem 1, we get the next inequalities:

$$\begin{aligned} p_0 &\geq \frac{2}{q_G N}(\epsilon_{pkipe} - \gamma \Pr[E_{msk}] - \Pr[F\text{-}Fail] - \Pr[D\text{-}Fail]), \\ p_1 &\geq \frac{1}{q_G N}(\epsilon_{pkipe} - \eta \Pr[\neg E_{msk}] - \Pr[F\text{-}Fail] - \Pr[D\text{-}Fail]), \end{aligned}$$

where $p_w := \Pr[\mathcal{A}_{cbdh}(g, g^a, g^b, g^c) = e(g,g)^{abc} | \mathcal{COIN} = w]$ for $w \in \{0,1\}$, and q_G, γ, η and E_{msk} are denoted similarly as in Theorem 1.

We then calculate $\Pr[F\text{-}Fail]$ and $\Pr[D\text{-}Fail]$. Since it is information theoretically impossible to obtain any informaion on R^*, $\mathcal{A}_{pkipe}$ submits R^* as in one of F-queries with probability at most $q_F/2^{\ell}$. $\mathcal{A}_{cbdh}$ fails to respond to a Decryption query only when $\mathcal{A}_{pkipe}$ succeeds to generate a ciphertext $C = \mathbf{Encrypt}(pk, \alpha, M; R)$ without submitting a F-query $\langle \alpha, M, R \rangle$. Hence, $\Pr[D\text{-}Fail]$ will be at most q_D/q.

Finally, by letting $\delta = 1/3$, we have $\epsilon_{cbdh} \geq \frac{2}{3q_G N}(\epsilon_{pkipe} - \frac{2q_F}{2^{\ell}} - \frac{2q_D}{q})$.

From the above discussions, we can easily see that the claimed bound of the running-time of $\mathcal{A}_{cbdh}$ holds. This completes the proof of the theorem. □

Strongly IND-KE-CCA Construction. We can build a strongly IND-KE-CCA scheme by combining the ideas of PKIPE2 and PKIPE3. A concrete construction of the scheme is given in the full version of this paper.

Acknowledgement

The authors would like to thank Nuttapong Attrapadung, Yang Cui, Yevgeniy Dodis, Jun Furukawa, Moti Yung, Rui Zhang, and anonymous referees for their comments and suggestions.

References

1. R. Anderson, "Two remarks on public key cryptology," Invited Lecture, ACM CCCS'97, available at `http://www.cl.cam.ac.uk/users/rja14/`.
2. M. Abdalla and L. Reyzin, "A new forward-secure digital signature scheme," Proc. of Asiacrypt'00, LNCS 1976, Springer-Verlag, pp. 116-129, 2000.
3. D. Boneh and X. Boyen, "Efficient selective-ID secure identity-based encryption without random oracles," Proc. of Eurocrypt'04, LNCS 3027, Springer-Verlag, pp.223-238, 2004.
4. D. Boneh and X. Boyen, "Secure identity based encryption without random oracles," Proc. of Crypto'04, LNCS 3152, Springer-Verlag, pp.443-459, 2004.
5. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," Proc. of Crypto'01, LNCS 2139, Springer-Verlag, pp.213-229, 2001.
6. D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," SIAM J. of Computing, vol. 32, no. 3, pp.586-615, 2003 (full version of [5]).
7. M. Bellare and A. Palacio, "Protecting against key exposure: strongly key-insulated encryption with optimal threshold," available at `http://eprint.iacr.org/2002/064/` .
8. M. Bellare and S.K. Miner, "A forward-secure digital signature scheme," Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp. 431-448, 1999.
9. C. Cocks, "An identity based encryption scheme based on quadratic residues," Proc. of IMA Int. Conf. 2001, Coding and Cryptography, LNCS 2260, Springer-Verlag, pp. 360-363, 2001.
10. R. Canetti, S. Halevi and J. Katz, "A forward secure public key encryption scheme," Proc. of Eurocrypt'03, LNCS 2656, Springer-Verlag, pp.255-271, 2003.
11. Y. Dodis, J. Katz, S. Xu and M. Yung, "Key-insulated public key cryptosystems," Proc. of Eurocrypt'02, LNCS 2332, Springer-Verlag, pp.65-82, 2002.
12. Y. Dodis, J. Katz, S. Xu and M. Yung, "Strong key-insulated signature schemes," Proc. of PKC'03, LNCS 2567, Springer-Verlag, pp.130-144, 2003.
13. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung, "Intrusion-resilient public-key encryption," Proc. of CT-RSA'03, LNCS 2612, Springer-Verlag, pp.19-32, 2003.
14. Y. Dodis, M. Franklin, J. Katz, A. Miyaji and M. Yung, "A generic construction for intrusion-resilient public-key encryption," Proc. of CT-RSA'04, LNCS 2964, Springer-Verlag, pp.81-98, 2004.
15. E. Fujisaki and T. Okamoto, "How to enhance the security of public-key encryption at minimum cost," Proc. of PKC'99, LNCS 1560, Springer-Verlag, pp.53-68, 1999.
16. E. Fujisaki and T. Okamoto, "Secure integration of asymmetric and symmetric encryption schemes," Proc. of Crypto'99, LNCS 1666, Springer-Verlag, pp.537-554, 1999.
17. G. Itkis and L. Reyzin, "SiBIR: signer-base intrusion-resilient signatures," Proc. of Crypto'02, LNCS 2442, Springer-Verlag, pp.499-514, 2002.
18. A. Shamir, "Identity-based cryptosystems and signature schemes," Proc. of Crypto'84, LNCS 196, Springer-Verlag, pp.47-53, 1985.
19. B. Waters, "Efficient identity based encryption without random oracles," Proc. of Eurocrypt'05, LNCS 3494, Springer-Verlag, pp.114-127, 2005.