Perfect Keyword Privacy in PEKS Systems

Mototsugu Nishioka

HITACHI, Ltd., Yokohama Research Laboratory, Japan
mototsugu.nishioka.rc@hitachi.com

Abstract. This paper presents a new security notion, called *perfect keyword privacy (PKP)*, for non-interactive public-key encryption with keyword search (PEKS) [5]. Although the conventional security notion for PEKS guarantees that a searchable ciphertext leaks no information about keywords, it gives no guarantee concerning leakage of a keyword from the trapdoor. PKP is a notion for overcoming this fatal deficiency. Since the trapdoor has verification functionality, the popular concept of "indistinguishability" is inadequate for capturing the notion of keyword privacy from the trapdoor. Hence, our formalization of PKP depends on the idea of formalizing a perfectly one-way hash function [10, 11]. We also present *IND-PKP security* as a useful notion for showing that a given PEKS scheme has PKP. Furthermore, we present PKP+ and IND-PKP+ as enhanced notions of PKP and IND-PKP, respectively. Finally, we present several instances of an IND-PKP or IND-PKP+ secure PEKS scheme, in either the random oracle model or the standard model.

1 Introduction

Much attention has been paid to encryption systems that go beyond traditional public-key encryption (PKE) systems, such as identity-based encryption (IBE) [6, 13, 17], public-key searchable encryption [5, 19], attribute-based encryption (ABE) [16], and functional encryption (FE) [7]. This paper deals with non-interactive public-key encryption with keyword search (PEKS), which is first presented in [5]. The PEKS provides a simple but useful mechanism to cryptographically protect data while keeping it available for search. For example, Alice can generate a searchable ciphertext corresponding to her selected keyword using Bob's public key. She then stores the ciphertext to a server. Bob can generate another key, called a *trapdoor*, corresponding to his selected keyword by using own secret key. Bob then sends the trapdoor to the server. The server can test whether or not the keywords corresponding to the ciphertext and the trapdoor are identical, and Bob can receive the ciphertext from the server only when the test is passed. In an email system, the server could be a gateway that forwards emails from Alice to Bob's portable terminal, depending on his selected keywords, such as "urgent" or "the next business meeting".

The conventional security for PEKS, called *IND-PEKS-CKA security* (cf. Definition 2), requires that the searchable ciphertext does not leak any information about the keyword. This security, however, gives no guarantee about leakage of the keyword from the trapdoor. Indeed, there exist PEKS schemes, such

as the statistical consistent scheme presented in [1], that are IND-PEKS-CKA secure but the trapdoor includes the keyword itself. This could bring serious problems in many systems. For instance, in the above example, the malicious server (or gateway) could collect the keywords selected by Bob from the given trapdoors and use them to analyze his activities. The privacy of keywords from the trapdoor has been discussed in the symmetric-key setting [14] and the interactive public-key setting [9]. On the other hand, to solve a similar problem in symmetric-key predicate encryption, Shen, Shi, and Waters [18] presented a security notion, *predicate privacy*, to ensure that tokens reveal no information about the encoded query predicate. Subsequently, Blundo, Iovino, and Persiano [4] presented a *predicate encryption scheme with partial public key*, and defined a *token security* to ensure the privacy of a pattern vector from a token. To the best of our knowledge, however, there has been no discussion of the leakage of keywords from trapdoors within the framework of PEKS, which is a non-interactive and "total" public key setting.

1.1 Contributions

This paper presents a new security notion for PEKS, called *perfect keyword privacy (PKP)*, to protect the privacy of a keyword from an adversary having both the trapdoor and the ciphertext of the underlying keyword. For formalizing PKP, the well-known concept of "indistinguishability" is inadequate. This is because a trapdoor has verification functionality; that is, when a keyword and trapdoor are given, one can easily verify whether the trapdoor corresponds to the keyword (see Section 3.1 for details). Therefore, we have applied the idea of formalizing a perfectly one-way hash function (POWHF) [10, 11].

Next, we present *IND-PKP security* as a useful notion for showing that a given PEKS scheme has PKP. The IND-PKP security can be defined in a game-based manner, whereas PKP is defined in a simulation-based manner. As compared with IND-PEKS-CKA security, IND-PKP security is a more extensive notion in the sense that it can ensure the privacy of a keyword from not only the ciphertext but also the trapdoor. Concerning the privacy of the keyword from only the ciphertext, however, IND-PKP security is a strictly weaker notion than IND-PEKS-CKA security. We demonstrate this by giving an instance of a PEKS scheme that is IND-PKP secure but not IND-PEKS-CKA secure (cf. Remark 7). Thus, PKP and IND-PKP security are independent notions from IND-PEKS-CKA security. Therefore, for higher security in PEKS, both IND-PEKS-CKA and IND-PKP securities are required. We also present PKP+ and IND-PKP+ security notions to enhance the PKP and IND-PKP security notions, respectively, from the viewpoint of *search pattern privacy*; that is, when two trapdoors are given, it is hard to guess whether they correspond to the same keyword.

Lastly, we give several instances of PEKS schemes that are IND-PKP secure or IND-PKP+ secure, in addition to being IND-PEKS-CKA secure. In Section 4.1, we describe the general methodology for constructing IND-PKP secure PEKS schemes. By using this methodology, in Section 4.2 we present a PEKS

scheme that is IND-PEKS-CKA and IND-PKP secure in the standard model. In Section 4.3, we present a PEKS scheme that is IND-PEKS-CKA and IND-PKP secure in the random oracle (RO) model, by direct construction. This scheme is based on the PEKS scheme in [5] and requires no computational assumptions for achieving IND-PKP security. In Section 4.4, we present a PEKS scheme that is IND-PEKS-CKA and IND-PKP+ secure in the RO model.

1.2 Related works

Numerous works on searchable encryption have been presented so far. In this section, we briefly describe only prior works that are specifically related to this paper. In particular, we concentrate on the public-key setting.

Boneh, Di Crescenzo, Ostrovsky, and Persiano [5] first presented the framework of PEKS. They formally defined its security and presented concrete schemes with this security. They also showed a general transformation from anonymous IBE to PEKS. Abdalla et al. [1] defined consistency in PEKS and gave an improved transformation from anonymous IBE to PEKS that guarantees consistency. They also introduced three extensions of the established notions: anonymous HIBE, PKE with temporary keyword search, and IBE with keyword search. Bellare, Boldyreva, and O'Neill [3] presented an efficiently searchable encryption (ESE) system to enable fast data search (i.e., logarithmic time in the database size) in outsourced databases. The ESE system utilizes a "tag", which can be generated in a deterministic manner both from the plaintext and from the corresponding ciphertext, as an index for search. Specifically, the server computes the tag of a ciphertext to be stored in the database and uses the tag to store the ciphertext appropriately in a data structure. The client computes and sends its tag to the server and receives any matches and associated data. They also presented an ESE scheme, called "Hash-and-Encrypt" encryption scheme, and showed that it is PRIV secure in the RO model when the underlying encryption scheme is IND-CPA secure. Unlike the trapdoor in the PEKS system, an ESE tag can be computed without a secret key. Therefore, ESE always allows data searches by anyone who can access the server. Moreover, its security depends on only the ciphertext, because the tag can be computed from it. Thus, the ESE system essentially has a different structure from that of PEKS, and it is outside the scope of this paper. Camenisch, Kohlweiss, Rial, and Sheedy [9] presented an extended notion of PEKS, called public-key encryption with oblivious keyword search (PEOKS), in which a user can obtain the trapdoor from the secret key holder without revealing the keyword. They constructed a PEOKS scheme by using a committed blind anonymous IBE scheme based on the anonymous IBE scheme in [8]. In PEOKS, however, the trapdoor is generated in an interactive manner. In contrast, our goal in this paper is to define and achieve security for guaranteeing the privacy of the keyword from the trapdoor, within the framework of PEKS (i.e., trapdoors are generated in a non-interactive manner). Boneh, Sahai, and Waters [7] presented a general framework for FE, and showed that existing encryption concepts, such as ABE and PE, can be expressed as particular functionalities of FE. They also discussed the formal security definition for

FE. They showed that the natural indistinguishability game-based definition is inadequate for certain functionalities since trivially insecure constructions may satisfy it. They hence presented a simulation-based security in which one getting the secret key reveals no information other than the result of decryption when the ciphertext is given. However, although simulation-based security can be achieved in the random oracle model, for a quite simple functionality (the functionality corresponding to IBE), it cannot be achieved even in the non-programmable random oracle model. Since PEKS can also be considered as a special case of FE, the security in [7] is applicable to PEKS. Both game-based security and simulation-based security, however, have the goal of achieving privacy of a keyword from a ciphertext, and they give no guarantee concerning keyword leakage from a trapdoor.

2 Preliminaries

We say that a function $f : \mathbb{N} \to [0,1]$ is *negligible* if, for every constant $c > 0$, there exists an integer k_c such that $f(k) \leq k^{-c}$ for all $k \geq k_c$. For a group G, G^* denotes a set $G \backslash \{1_G\}$, where 1_G is an identity element of G. For a finite set S, $x \leftarrow S$ denotes the operation of picking an element uniformly from S. We use $x, x' \leftarrow S$ as shorthand for $x \leftarrow S\ ;\ x' \leftarrow S$. If A is a probabilistic algorithm, then $y \leftarrow A(x_1, x_2, \ldots; r)$ is the result of running A on inputs $x_1, x_2, \ldots$ and coins r. We let $y \leftarrow A(x_1, x_2, \ldots)$ denote the experiment of picking r at random and letting y to be $A(x_1, x_2, \ldots; r)$. The notation $\Pr[x_1 \leftarrow S_1;\ x_2 \leftarrow S_2;\ \ldots : p(x_1, x_2, \ldots)]$ denotes the probability that the predicate $p(x_1, x_2, \ldots)$ is true after the ordered execution of $x_1 \leftarrow S_1,\ x_2 \leftarrow S_2$, and so on. If α is neither an algorithm nor a set then $x \leftarrow \alpha$ is a simple assignment statement. For a random variable $\mathbf{X}$, $[\mathbf{X}]$ denotes a set $\{x \mid \Pr[\mathbf{X} = x] > 0\}$, and $||\mathbf{X}||$ denotes a value $\max_{x \in [\mathbf{X}]}\{\Pr[\mathbf{X} = x]\}$. $E(\mathbf{X})$ denotes the *expectation* of $\mathbf{X}$, and $x \leftarrow \mathbf{X}$ denotes selection of a random sample from $\mathbf{X}$; thus, $\Pr[x \leftarrow \mathbf{X}] = \Pr[\mathbf{X} = x]$. We use $x, x' \leftarrow \mathbf{X}$ as shorthand for $x \leftarrow \mathbf{X}\ ;\ x' \leftarrow \mathbf{X}$. The random variables $\mathbf{X}$ and $\mathbf{Y}$ are *independent* if $\Pr[\mathbf{X} = a \wedge \mathbf{Y} = b] = \Pr[\mathbf{X} = a] \cdot \Pr[\mathbf{Y} = b]$ for any $a, b \in \{0,1\}^*$. A *probability ensemble* is a sequence $\mathcal{X} = \{\mathbf{X}_k\}_{k \in \mathbb{N}}$ of random variables $\mathbf{X}_k$. We say that $\mathcal{X}$ is *well-spread* if $||\mathbf{X}_k||$ is negligible in k. The *d-composite bilinear group generator* $\mathcal{G}$ is a PPT algorithm that takes a security parameter k as input and outputs $(p_1, \ldots, p_d, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$, where p_i are prime numbers with $p_i > 2^k$, $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups with order $N = \prod_{i=1}^{d} p_i$, and $\mathbf{e}$ is a map from $\mathbb{G} \times \mathbb{G}$ to $\mathbb{G}_T$, called a *bilinear map*, with the following properties:

1. Computable: There is an efficient algorithm to compute $\mathbf{e}(g, h)$ for any $g, h \in \mathbb{G}$.
2. Bilinear: $\mathbf{e}(g^x, g^y) = \mathbf{e}(g, g)^{xy}$ for any $g \in \mathbb{G}$ and any $x, y \in \mathbb{Z}_N$.
3. Non-degenerate: If g is a generator of $\mathbb{G}$ then $\mathbf{e}(g, g)$ is a generator of $\mathbb{G}_T$.

In particular, the 1-composite bilinear group generator is simply called a *bilinear group generator*. For an integer m dividing N, $\mathbb{G}_m$ denotes the subgroup of $\mathbb{G}$ with order m. Then, $\mathbf{e}(x, y) = 1_{\mathbb{G}}$ for any $x \in \mathbb{G}_m$ and any $y \in \mathbb{G}_n$ when m and n are coprime. This is called the "orthogonality property".

Definition 1. *A* non-interactive public-key encryption with keyword search (PEKS) *scheme consists of the following polynomial-time randomized algorithms:*

- $\mathsf{KG}(1^k)$*: Takes a security parameter* k*, and generates a public/secret key pair* (PK, SK)*. Here, the keys include the information about the keyword space* KSP_k*.*
- $\mathsf{Td}(SK, w)$*: For* SK *and a keyword* $w \in \mathsf{KSP}_k$*, produces a* trapdoor T_w*.*
- $\mathsf{PEKS}(PK, w)$*: For* PK *and* $w \in \mathsf{KSP}_k$*, produces a* searchable ciphertext C_w *of* w*.*
- $\mathsf{Test}(PK, C_w, T_{w'})$*: For* PK*,* $C_w = \mathsf{PEKS}(PK, w)$*, and* $T_{w'} = \mathsf{Td}(SK, w')$*, where* $w, w' \in \mathsf{KSP}_k$*, outputs* 1 *if* $w = w'$*. Otherwise, outputs* 0 *with an overwhelming probability*[1]*.*

The security of PEKS is defined against an active attacker who is able to obtain a trapdoor T_w for any keyword w of his choice, to ensure that a $\mathsf{PEKS}(PK, w)$ does not reveal any information about w *unless* T_w *is available* [5].

IND-PEKS-CKA security. Let $\Pi = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ be a PEKS scheme, and let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be a probabilistic polynomial-time (PPT) adversary. We then consider the following experiment.

Experiment $\mathsf{Exp}^{\mathrm{ind:peks}}_{\mathcal{A},\Pi}(k)$
$(PK, SK) \leftarrow \mathsf{KG}(1^k)$; $(w_0, w_1, \sigma) \leftarrow \mathcal{A}_1^{\mathsf{Td}(SK,\cdot)}(1^k, PK)$
$b \leftarrow \{0,1\}$; $C_{w_b} \leftarrow \mathsf{PEKS}(PK, w_b)$; $b' \leftarrow \mathcal{A}_2^{\mathsf{Td}(SK,\cdot)}(1^k, PK, \sigma, w_0, w_1, C_{w_b})$
If $b = b'$ then return 1 else return 0.

Here, $w_0, w_1 \in \mathsf{KSP}_k$ and $w_0 \neq w_1$, σ is a string representing the configuration of $\mathcal{A}_1$ at its quitting point, and $\mathcal{A}$ is prohibited from asking for the trapdoors w_0 or w_1. The *advantage* of $\mathcal{A}$ in the above experiment is defined as

$$\mathsf{Adv}^{\mathrm{ind:peks}}_{\mathcal{A},\Pi}(k) = \left| \Pr\left[\mathsf{Exp}^{\mathrm{ind:peks}}_{\mathcal{A},\Pi}(k) = 1\right] - \frac{1}{2} \right|.$$

Definition 2. *We say that a PEKS scheme* Π *is* indistinguishable against a chosen-keyword attack (CKA)*, briefly,* IND-PEKS-CKA secure*, if* $\mathsf{Adv}^{\mathrm{ind:peks}}_{\mathcal{A},\Pi}(k)$ *is negligible for any* $\mathcal{A}$*.*

3 Perfect keyword privacy

3.1 Definition

The IND-PEKS-CKA security (in Definition 2) guarantees the privacy of the keyword from a searchable ciphertext. It does not, however, guarantee any security concerning leakage of the keyword from the trapdoor. For example, in [1], a PEKS scheme with statistical consistency is presented and shown to be

[1] This property is called *computational consistency* in [1]. In this paper, we call it "consistency" for brevity.

IND-PEKS-CKA secure under the BDH assumption. That scheme is designed, however, so that the trapdoor includes the keyword itself. To overcome this deficiency, we present a new security notion, *perfect keyword privacy* (briefly, PKP), for a PEKS to ensure the privacy of the keyword from both the trapdoor and the searchable ciphertext. In this section, we present a formal definition of PKP. In formulating security against information leakage, the natural, popular concept that comes to mind is "indistinguishability". We first explain why indistinguishability is inadequate for defining PKP. We now consider the following game based on indistinguishability.

1. For $(PK, SK) \leftarrow \mathsf{KG}(1^k)$, the adversary receives the public key PK and is allowed to access to the trapdoor oracle $\mathsf{Td}(SK, \cdot)$.
2. In the challenge phase, the adversary submits two keywords, w_0, w_1, and receives a target trapdoor $T_{w_b} = \mathsf{Td}(SK, w_b)$ for a randomly chosen $b \in \{0, 1\}$. The adversary can continuously make queries to the trapdoor oracle $\mathsf{Td}(SK, \cdot)$, except for querying w_0 or w_1.
3. In the guess phase, the adversary finally outputs $b' \in \{0, 1\}$ as its guess for b.

It is then required that no PPT adversary can guess the challenge bit b with a non-negligible advantage. There exists an adversary, however, that can guess b with an overwhelming probability in the above game. After receiving the trapdoor T_{w_b} in Step 2, the adversary computes $C_{w_i} = \mathsf{PEKS}(PK, w_i)$ for each $i = 0, 1$ and outputs $b' \in \{0, 1\}$ such that $\mathsf{Test}(T_{w_b}, C_{w_{b'}}) = 1$. Then, from the consistency of PEKS, the probability $\Pr[b = b']$ is overwhelming.

Our formalization of PKP depends on an idea of formalizing a POWHF [10, 11]. Informally, we say that a PEKS scheme has PKP if there is no efficient way to guess the keyword w from the given trapdoor T_w and ciphertext C_w other than the "select and test" method; in other words, the adversary selects a keyword w' in an arbitrary manner and tests whether $\mathsf{Test}(T_w, \mathsf{PEKS}(PK, w')) = 1$ holds. If the test is passed, the adversary decides that $w = w'$. In our definition, the "select and test" method is formalized by an oracle $\mathcal{O}_w$, called a *test oracle*, in the *ideal system*: for a query (keyword) w', $\mathcal{O}_w$ responds with 1 if $w = w'$; otherwise, it responds with 0. Note that one may think that the oracle $\mathcal{O}_w$ should be defined so that it outputs 0 with an overwhelming probability when $w \neq w'$ because Definition 1 adopts computational consistency. It can easily be shown, however, that this difference does not affect Definition 3.

Perfect keyword privacy. Let $\Pi = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ be a PEKS scheme. Let $\mathcal{X} = \{\mathbf{X}_k\}_{k \in \mathbb{N}}$ be a probability ensemble such that $[\mathbf{X}_k] = \mathsf{KSP}_k$. From now on, unless otherwise indicated, we assume that $\mathcal{X}$ is well-spread and independent from key generation (cf. Remarks 1 and 2). $\mathcal{X}$ determines the distribution of keywords; that is, when the security parameter k is given, the keyword w is given as a random sample from $\mathbf{X}_k$. Let $\mathcal{P} = \{P_k\}_{k \in \mathbb{N}}$ be a predicate family, where P_k is an efficiently computable predicate over $[\mathbf{X}_k]$. Let $\mathcal{A}$ and $\mathcal{B}$ be PPT algorithms. We then define the following experiments. See Section 2 for other notations and conventions.

Experiment $\mathsf{Exp}^{\mathrm{pkp:real}}_{\mathcal{A},\Pi,\mathcal{X},\mathcal{P}}(k)$	**Experiment** $\mathsf{Exp}^{\mathrm{pkp:ideal}}_{\mathcal{B},\Pi,\mathcal{X},\mathcal{P}}(k)$
$w \leftarrow \mathbf{X}_k$; $(PK, SK) \leftarrow \mathsf{KG}(1^k)$	$w \leftarrow \mathbf{X}_k$; $(PK, SK) \leftarrow \mathsf{KG}(1^k)$
$T_w \leftarrow \mathsf{Td}(SK, w)$; $C_w \leftarrow \mathsf{PEKS}(PK, w)$	$z \leftarrow \mathcal{B}^{\mathcal{O}_w, \mathsf{Td}(SK,\cdot)}(1^k, PK)$
$z \leftarrow \mathcal{A}^{\mathsf{Td}(SK,\cdot)}(1^k, PK, T_w, C_w)$	If $z = P_k(w)$ then return 1
If $z = P_k(w)$ then return 1 else return 0.	else return 0.

Definition 3. *We say that a PEKS scheme Π has* perfect keyword privacy (PKP) with respect to $\mathcal{X}$ *if for any $\mathcal{P}$ and $\mathcal{A}$, there exists a negligible function* negl *and $\mathcal{B}$ such that*

$$\Pr\left[\mathsf{Exp}^{\mathrm{pkp:real}}_{\mathcal{A},\Pi,\mathcal{X},\mathcal{P}}(k) = 1\right] \leq \Pr\left[\mathsf{Exp}^{\mathrm{pkp:ideal}}_{\mathcal{B},\Pi,\mathcal{X},\mathcal{P}}(k) = 1\right] + \mathsf{negl}(k) \tag{1}$$

for all $k \in \mathbb{N}$. We also say that Π has PKP *if it has PKP with respect to any $\mathcal{X}$.*

Remark 1. In Definition 3, the probability ensemble $\mathcal{X}$ is given independently from the key generation of the PEKS scheme. This setting is very significant for obtaining a useful notion, *IND-PKP security*, to achieve PKP (see the proof of Theorem 1). From a practical viewpoint, we think that this is a natural setting in the real world. Generally, public keys are not used as keywords because they are large, meaningless phrases, whereas other identifiers, such as a user's name and email address, are usually used to designate a person.

Remark 2. Definition 3 is meaningful even if $\mathcal{X}$ is not well-spread. However, without loss of generality, we can assume that the probability ensemble $\mathcal{X}$ is well-spread when defining the privacy of the keyword from the trapdoor. As described at the beginning of this section, if the trapdoor is given, the adversary can always verify whether it corresponds to his own chosen keyword. From this fact, in (1) we can exclude the case of choosing $w \in [\mathbf{X}_k]$ such that $\Pr[\mathbf{X}_k = w]$ is non-negligible. Notice that the number of keywords appearing with a non-negligible probability is polynomially bounded in k.

Remark 3. In Definition 3, only a single tuple of the trapdoor and ciphertext is given to the adversary $\mathcal{A}$. In Section 3.2, we present a notion, IND-PKP security, and use it to show that a given PEKS scheme has PKP. From a hybrid argument [2], we can show that (single-target) IND-PKP security implies multi-target IND-PKP security. Thus, IND-PKP security implies multi-target PKP.

Remark 4. Concerning the privacy of a keyword from only the searchable ciphertext, IND-PEKS-CKA security gives strictly stronger security than that of PKP. In Remark 7, we demonstrate this by presenting a PEKS scheme that is IND-PKP secure (cf. Section 3.2) but not IND-PEKS-CKA secure. On the other hand, there exist PEKS schemes, such as the scheme in [1] described above, that are IND-PEKS-CKA secure but do not have PKP. Thus, PKP is a separate security notion from IND-PEKS-CKA security; that is, PKP and IND-PEKS-CKA security are independent of each other. Hence, for higher security in a PEKS system, both IND-PEKS-CKA security and PKP are required. Note that strictly speaking, the above results on separation and comparison are true under some computational complexity assumptions because they are required for achieving the securities of the instances.

We expect that the idea of PKP will be applied in FE systems to ensure the privacy of a key from a secret key (see [7] for the detail of FE); since FE is a generalized concept of many other primitives, such as IBE, PE, and ABE, this idea is also applicable to those primitives. Informally, we say that an FE scheme for a functionality F over (K, X) has *perfect key privacy* if a secret key sk_k corresponding to the key $k \in K$ leaks no information about k, beyond the information obtained from the oracle $\mathcal{O}_{F(k,\cdot)}$, where for the query x, $\mathcal{O}_{F(k,\cdot)}$ returns $F(k, x)$. If $\mathcal{O}_{F(k,\cdot)}$ gives only trivial information[2], like $\mathcal{O}_x$ in PEKS, then this notion will give meaningful security in an FE system. We leave a detailed, formal discussion to subsequent works.

3.2 How to achieve PKP

In this section, we present a useful notion, called *IND-PKP security*, to show that a given PEKS scheme has PKP. The IND-PKP security can be defined in a game-based manner, whereas we defined PKP above in a simulation-based manner. The IND-PKP security can be regarded as a strictly stronger notion than PKP from the viewpoint of the strength relation between the cryptographic assumptions for achieving these securities (cf. Remark 6).

IND-PKP security. Let $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ be a probability ensemble, and let $\Pi = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ be a PEKS scheme. Let $\mathcal{A}$ be a PPT algorithm, called *IND-PKP adversary*. We then define the following experiment (cf. Remark 3).

Experiment $\mathsf{Exp}^{\text{ind-pkp}}_{\mathcal{A},\Pi,\mathcal{X}}(k)$
$w_0, w_1 \leftarrow \mathbf{X}_k$; $b \leftarrow \{0,1\}$; $(PK, SK), (PK', SK') \leftarrow \mathsf{KG}(1^k)$
$T_{w_0} \leftarrow \mathsf{Td}(SK, w_0)$; $C_{w_0} \leftarrow \mathsf{PEKS}(PK, w_0)$
$T'_{w_b} \leftarrow \mathsf{Td}(SK', w_b)$; $C'_{w_b} \leftarrow \mathsf{PEKS}(PK', w_b)$
$b' \leftarrow \mathcal{A}^{\mathsf{Td}(SK,\cdot),\mathsf{Td}(SK',\cdot)}(1^k, PK, T_{w_0}, C_{w_0}, PK', T'_{w_b}, C'_{w_b})$
If $b = b'$ then return 1 else return 0.

The *advantage* of $\mathcal{A}$ in the above experiment is defined as

$$\mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A},\Pi,\mathcal{X}}(k) = \left| \Pr\left[\mathsf{Exp}^{\text{ind-pkp}}_{\mathcal{A},\Pi,\mathcal{X}}(k) = 1\right] - \frac{1}{2} \right|,$$

and $b \in \{0,1\}$ is called a *challenge bit*.

Definition 4. *We say that a PEKS scheme Π is* IND-PKP secure with respect to $\mathcal{X}$ *if* $\mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A},\Pi,\mathcal{X}}(k)$ *is negligible for any $\mathcal{A}$. We also say that Π is* IND-PKP secure *if it is IND-PKP secure with respect to any $\mathcal{X}$.*

Theorem 1. *If the PEKS scheme $\Pi = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ is IND-PKP secure, then it has PKP.*

The proof of Theorem1 is given in Appendix A.

[2] For example, $F(k, x)$ represents a result of execution of certain program P_k for input x. P_k outputs a meaningful string only for particular x, whereas it outputs $\perp$ for other input. It is easy to find such particular x from k but difficult to find it from sk_k.

3.3 Additional notions

In Section 4.3, we present an IND-PKP secure PEKS system in which the trapdoor is generated in a deterministic manner. In this system, when two trapdoors are given under the same secret key, one can easily guess whether they correspond to the same keyword. Thus, IND-PKP security cannot assure "search pattern privacy", in general. In this section, we address this issue.

Search pattern privacy. Let $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ be a probability ensemble, and let $\Pi = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ be a PEKS scheme. Let $\mathcal{A}$ be a PPT algorithm, called a *SPP adversary*. We then define the following experiment.

Experiment $\mathsf{Exp}^{\mathrm{spp}}_{\mathcal{A},\Pi,\mathcal{X}}(k)$
$w_0, w_1 \leftarrow \mathbf{X}_k$; $b \leftarrow \{0,1\}$; $(PK, SK) \leftarrow \mathsf{KG}(1^k)$
$T_{w_0} \leftarrow \mathsf{Td}(SK, w_0)$; $T_{w_b} \leftarrow \mathsf{Td}(SK, w_b)$; $b' \leftarrow \mathcal{A}^{\mathsf{Td}(SK,\cdot)}(1^k, PK, T_{w_0}, T_{w_b})$
If $b = b'$ then return 1 else return 0.

The *advantage* of $\mathcal{A}$ in the above experiment is defined as

$$\mathsf{Adv}^{\mathrm{spp}}_{\mathcal{A},\Pi,\mathcal{X}}(k) = \left| \Pr\left[\mathsf{Exp}^{\mathrm{spp}}_{\mathcal{A},\Pi,\mathcal{X}}(k) = 1\right] - \frac{1}{2} \right|,$$

and $b \in \{0,1\}$ is called a *challenge bit*.

Definition 5. *We say that a PEKS scheme Π has* search pattern privacy *(briefly, SPP) if $\mathsf{Adv}^{\mathrm{spp}}_{\mathcal{A},\Pi,\mathcal{X}}(k)$ is negligible for any $\mathcal{A}$ and $\mathcal{X}$.*

Definition 6 (PKP+ and IND-PKP+). *We say that a PEKS scheme has* PKP+ *if it has both PKP and SPP. We also say that a PEKS scheme is* IND-PKP+ secure *if it is IND-PKP secure and has SPP.*

Remark 5. In Definition 5, it is essential that the adversary cannot see the ciphertexts C_{w_0} and C_{w_1}. If either of these is given, the adversary can easily guess b by running the test algorithm. Thus, in a real system, SPP is meaningful in a situation in which there is no ciphertext corresponding to the search keyword (although the searcher has multiple trapdoors corresponding to the underlying keyword). In our definition of SPP, the adversary is not allowed to choose the keywords w_0, w_1. This is because we regard SPP as an additional notion for PKP to strengthen the privacy of keywords.

4 PEKS schemes with perfect keyword privacy

As described in Remark 4, concerning the privacy of a keyword from only a searchable ciphertext, IND-PKP security ensures strictly weaker security than that of IND-PEKS-CKA security. Therefore, for higher security in PEKS, we present several instances of a PEKS scheme that is IND-PKP secure or IND-PKP+ secure, in addition to being IND-PEKS-CKA secure. As much as we possible, we looked for appropriate instances in existing schemes and modified them if necessary.

4.1 General methodology

Before giving concrete instances, we describe a general methodology for achieving IND-PKP security in PEKS schemes. We first introduce the notion of a *secure injective-function generator*.

Definition 7. *The* injective-function generator *is a pair of PPT algorithms $\mathcal{I}$ and $\mathfrak{G}$ such that (1) $\mathcal{I}$ takes a security parameter k as input and outputs $\lambda_k \in \{0,1\}^*$, and (2) $\mathfrak{G}$ takes λ_k as input and outputs an injective function $\pi : Y_{\lambda_k} \to Z_{\lambda_k}$, where Y_{λ_k} and Z_{λ_k} are sets uniquely determined from λ_k. We say that the injective-function generator $(\mathcal{I}, \mathfrak{G})$ is* secure *if for any well-spread probability ensemble $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ with $[\mathbf{X}_k] \subseteq Y_{\lambda_k}$, and any PPT algorithm $\mathcal{B}$,*

$$\mathsf{Adv}^{\mathrm{sif}}_{\mathcal{B},\mathcal{I},\mathfrak{G},\mathcal{X}}(k) = \Bigg|\Pr\Big[\lambda_k \leftarrow \mathcal{I}(1^k)\ ;\ \pi,\pi' \leftarrow \mathfrak{G}(\lambda_k)\ ;\ x_0,x_1 \leftarrow \mathbf{X}_k\ ;\ b \leftarrow \{0,1\}\ ;\ b' \leftarrow \mathcal{B}(1^k,\pi,\pi',\pi(x_0),\pi'(x_b))\ :\ b=b'\Big] - \frac{1}{2}\Bigg|$$

is negligible.

An example of a secure injective-function generator is given in Section 4.2. Next, we describe how to convert a PEKS scheme into an IND-PKP secure PEKS scheme by using a secure injective-function generator. The essential point of the conversion is that the secure function generator yields a fresh injective function for each user, and the trapdoor and ciphertext are created from the keyword's function value. Let $\Pi = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ be a PEKS scheme, and let $(\mathcal{I}, \mathfrak{G})$ be an injective-function generator such that for $\lambda_k \leftarrow \mathcal{I}(1^k)$, $\mathfrak{G}(\lambda_k)$ outputs an injective function from KSP_k to KSP_k. We then define a PEKS scheme $\Pi^* = (\mathsf{KG}^*, \mathsf{Td}^*, \mathsf{PEKS}^*, \mathsf{Test}^*)$ as follows.

- $\mathsf{KG}^*(1^k)$ outputs $(PK^*, SK^*) = ((PK, \lambda_k, \pi), (SK, \lambda_k, \pi))$ for $(PK, SK) \leftarrow \mathsf{KG}(1^k)$, $\lambda_k \leftarrow \mathcal{I}(1^k)$, and $\pi \leftarrow \mathfrak{G}(\lambda_k)$, where λ_k is a common parameter for all users in this system.
- $\mathsf{Td}^*(SK^*, w)$ outputs $T_{\pi(w)} \leftarrow \mathsf{Td}(SK, \pi(w))$.
- $\mathsf{PEKS}^*(PK^*, w)$ outputs $C_{\pi(w)} \leftarrow \mathsf{PEKS}(PK, \pi(w))$.
- Test^* is identical with Test.

Theorem 2. *In the PEKS scheme Π^*, we have the following results.*

(a) *If $(\mathcal{I}, \mathfrak{G})$ is secure, then Π^* is IND-PKP secure.*
(b) *If Π is IND-PEKS-CKA secure, then Π^* is IND-PEKS-CKA secure.*

The proof of Theorem 2 is given in Appendix B. The above methodology is simple and useful although some additional assumption may be required for secure function generator. This methodology however cannot guarantee SPP in Π^*. The brute force approach (under a constraint) for obtaining an IND-PKP+ secure PEKS scheme Π^* by using a secure injective-function generator $(\mathcal{I}, \mathfrak{G})$ is as follows. KG^* creates $(PK^*, SK^*) = ((PK, \lambda_k, \pi_1, \ldots, \pi_n), (SK, \lambda_k, \pi_1, \ldots, \pi_n))$

for $(PK, SK) \leftarrow \mathsf{KG}(1^k)$, $\lambda_k \leftarrow \mathcal{I}(1^k)$, and $\pi_1, \ldots, \pi_n \leftarrow \mathfrak{G}(\lambda_k)$. $\mathsf{Td}^*(SK, w)$ has a counter, and it outputs $T_w^* = T_{\pi_i(w)}$ if this is the i-th execution for the same keyword w. $\mathsf{PEKS}^*(PK, w)$ outputs $C_w^* = (C_{\pi_i(w)})_{1 \le i \le n}$. It can readily be shown that if $(\mathcal{I}, \mathfrak{G})$ is a secure injective-function generator and the adversary is restricted to making at most n trapdoor queries to the same keyword, then Π^* is IND-PKP+ secure. We do not know of a general methodology for obtaining an IND-PKP+ secure PEKS scheme without restriction. This problem remains open. Note that obviously, we can obtain a similar result to Theorem 2 when applying an RO generator (i.e., π is an RO in the above Π^*) instead of a secure injective-function generator (cf. Proposition 5).

4.2 Instance 1

In this section, we present a concrete instance of a PEKS scheme that can be obtained by the methodology described in Section 4.1. This instance is based on the Gentry IBE scheme [15] and the conversion [1] from the IBE scheme to the PEKS scheme. The resulting scheme is both IND-PKP and IND-PEKS-CKA secure in the standard model. Let $\mathcal{G}$ be a bilinear group generator. We define an injective function generator $(\mathcal{G}, \mathfrak{G})$ as follows: For $I = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^k)$, $\mathfrak{G}(I)$ picks a primitive element $\xi \in \mathbb{Z}_p^*$ at random and outputs a function π such that $\pi(x) = \xi^x$ for $x \in \mathbb{Z}_p$. Since ξ is a primitive element, π is injective. The following assumption can be seen as a variant of the Decisional Diffie-Hellman (DDH) assumption.

Assumption I We say that $\mathcal{G}$ satisfies Assumption I if for any well-spread probability ensemble $\mathcal{X} = \{\mathbf{X}_k\}_{k \in \mathbb{N}}$ with $[\mathbf{X}_k] \subseteq \mathbb{Z}_p$, and any PPT algorithm $\mathcal{B}$,

$$\Big|\Pr[I = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^k)\ ;\ x_0, x_1 \leftarrow \mathbf{X}_k\ ;\ \xi_1, \xi_2 \leftarrow \mathsf{PRIM}(p)\ ; \\ b \leftarrow \{0,1\}\ ;\ b' \leftarrow \mathcal{B}(1^k, I, \xi_1, \xi_2, \xi_1^{x_0}, \xi_2^{x_b})\ :\ b = b'] - \frac{1}{2}\Big|$$

is negligible, where $\mathsf{PRIM}(p)$ is a set of all primitive elements in $\mathbb{Z}_p$.

From the definitions, the following proposition is clear.

Proposition 1. *If $\mathcal{G}$ satisfies Assumption I, then $(\mathcal{G}, \mathfrak{G})$ is secure.*

Let $(\mathcal{G}, \mathfrak{G})$ be the injective-function generator mentioned above. We then define the PEKS scheme $\Pi_1 = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ as follows.

- $\mathsf{KG}(1^k)$: For a security parameter k, run $\mathcal{G}$ and $\mathfrak{G}$ to obtain $I = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^k)$ and $\pi(x) = \xi^x \leftarrow \mathfrak{G}(I)$. Pick $g, h \in \mathbb{G}^*$ and $\alpha \in \mathbb{Z}_p$ at random, set $PK = (I, \pi, g, g_1 = g^\alpha, h)$ and $SK = \alpha$, and output (PK, SK), where (I, g) are common parameters for all users in this system.
- $\mathsf{Td}(SK, w)$: To generate a trapdoor for a keyword $w \in \mathbb{Z}_p$ under the secret key SK, pick a random $r_w \in \mathbb{Z}_p$ and output $T_w = (r_w, h_w = (hg^{-r_w})^{\frac{1}{\alpha - \pi(w)}})$. Note that the same r_w is used for the same keyword w.

– $\mathsf{PEKS}(PK, w)$: To encrypt a keyword w under the public key PK, pick random $s \in \mathbb{Z}_p$ and $R \in \mathbb{G}_T$, and output $C_w = (R,\ C_1 = g_1^s g^{-s\pi(w)},\ C_2 = \mathbf{e}(g,g)^s,\ C_3 = R \cdot \mathbf{e}(g,h)^{-s})$.
– $\mathsf{Test}(PK, T_w, C_w)$: Using the notation in the description of Td and PEKS, if $R = C_3 \cdot \mathbf{e}(C_1, h_w) C_2^{r_w}$ then output 1; otherwise, output 0.

The Gentry IBE scheme is shown to be anonymous and IND-ID-CPA secure under the truncated decision ABDHE assumption (see [15] for details). Therefore, from Theorem 4.2 in [1] and Theorem 2. (b), Π_1 is IND-PEKS-CKA secure under the same assumption, and it is computationally consistent. In addition, from Proposition 1 and Theorem 2. (a), Π_1 is IND-PKP secure under Assumption I. However, Π_1 is not IND-PKP+ secure because the trapdoor in Π_1 is uniquely determined per the keyword. An instance of a PEKS scheme that is both IND-PEKS-CKA and IND-PKP+ secure is given in Section 4.4.

4.3 Instance 2

In this section, we present an efficient PEKS scheme that is IND-PKP and IND-PEKS-CKA secure in the RO model, without depending on a secure injective-function generator in its construction. To achieve IND-PKP security, this instance requires no cryptographic assumption beyond those for achieving IND-PEKS-CKA security. This scheme is based on the PEKS scheme proposed in [5], with slight modification. Let $\mathcal{G}$ be a bilinear group generator. We begin by describing the PEKS scheme $\Pi_2 = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ associated with $\mathcal{G}$.

– $\mathsf{KG}(1^k)$: For a security parameter k, run $\mathcal{G}$ to obtain $(p, \mathbb{G}, \mathbb{G}_1, \mathbf{e}) \leftarrow \mathcal{G}(1^k)$, and select $a \in \mathbb{Z}_p$ and $g \in \mathbb{G}^*$ (i.e., g is a generator of $\mathbb{G}$) at random. Set $PK = (p, g, \mathbb{G}, \mathbb{G}_1, \mathbf{e}, g, h = g^a, H_1, H_2)$ and $SK = (PK, a)$, where H_1 and H_2 are hash functions, and $(p, \mathbb{G}, \mathbb{G}_1, \mathbf{e}, g, H_1, H_2)$ are common parameters for all users in this system, and output (PK, SK).
– $\mathsf{Td}(SK, w)$: As a trapdoor for a keyword $w \in \{0,1\}^*$ under the secret key $SK = (PK, a)$, output $T_w = H_1(PK||w)^a \in \mathbb{G}$.
– $\mathsf{PEKS}(PK, w)$: To encrypt a keyword w under the public key PK, pick a random $r \in \mathbb{Z}_p$ and output $C_w = (C_1 = g^r,\ C_2 = H_2(\mathbf{e}(H_1(PK||w), h^r)))$.
– $\mathsf{Test}(PK, C_w, T_w)$: Using the notation in the description of Td and PEKS, if $H_2(\mathbf{e}(T_w, C_1)) = C_2$, then output 1; otherwise, output 0.

The consistency of the above PEKS scheme is shown in [1]. As compared to the original scheme, the input to H_1 includes a public key in Π_2. This modification does not collapse the IND-PEKS-CKA security of the scheme because it can be seen as the original PEKS scheme with a special keyword form. Therefore, like the original scheme, this scheme can be shown IND-PEKS-CKA secure in the RO model under the BDH assumption [5]. Interestingly, IND-PKP security of Π_2 can be shown only under the RO assumption (i.e., without a computational assumption).

Proposition 2. *Suppose that H_1 and H_2 are ROs. For any probability ensemble $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ and any IND-PKP adversary $\mathcal{A}$ against Π_2 that makes at most $q_{H_1}(k)$ queries to H_1 and at most $q_t(k)$ trapdoor queries when the security parameter k is given,*

$$\mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A},\Pi_2,\mathcal{X}}(k) \leq 2(q_t(k) + q_{H_1}(k)) \cdot ||\mathbf{X}_k|| + 2^{-k} \quad (k \in \mathbb{N}).$$

The proof of Proposition 2 is given in Appendix C.

Remark 6. From Definition 3, the original PEKS scheme in [5] is shown directly to has a PKP under the RO assumption. This is because in the trapdoor, the keyword is hidden by the RO H_1, and the ciphertexts can be created from the trapdoor. However, it will be impossible to show the IND-PKP security of this scheme only under the RO assumption. If the DDH assumption (on $\mathbb{G}$) is added, then the scheme is shown to be IND-PKP secure. In this sense, IND-PKP security can be regarded as a strictly stronger notion than PKP. On the other hand, if H_1 and H_2 are freshly chosen in each key generation (not used as common parameters), then the original scheme is shown to be IND-PKP secure only under the RO assumption.

4.4 Instance 3

As described in Section 3.1, achievement of both IND-PEKS-CKA and IND-PKP+ securities can be considered as the highest security in a PEKS system. Unfortunately, we could not find an appropriate instance within any existing schemes (even allowing for slight modification). We then present a new PEKS scheme that is IND-PKP+ and IND-PEKS-CKA secure in the RO model. Let $\mathcal{G}_3$ be a 3-composite bilinear group generator. We begin by describing the PEKS scheme $\Pi_3 = (\mathsf{KG}, \mathsf{Td}, \mathsf{PEKS}, \mathsf{Test})$ associated with $\mathcal{G}_3$.

- $\mathsf{KG}(1^k)$: For a security parameter k, run $\mathcal{G}_3$ to obtain $I = (p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}_3(1^k)$ and set $N = p_1p_2p_3$. Pick $g_i \in \mathbb{G}^*_{p_i}$ $(1 \leq i \leq 3)$ and $R_2 \in \mathbb{G}_{p_2}$ at random. Set $PK = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_2, g_3, g{=}g_1R_2, H)$ and $SK = (PK, g_1)$, where H is a hash function from $\{0,1\}^*$ to $\mathbb{G}_{p_1}$, and output (PK, SK).
- $\mathsf{Td}(SK, w)$: To generate a trapdoor of a keyword $w \in \{0,1\}^*$ under the secret key SK, pick $s \in \mathbb{Z}_{p_1}$ and $R_3, S_3 \in \mathbb{G}_{p_3}$ at random, and output $T_w = (T_1 = g_1^s R_3,\ T_2 = H(w)^s S_3)$.
- $\mathsf{PEKS}(PK, w)$: To encrypt a keyword w under the public key PK, pick $r \in \mathbb{Z}_N$ and $Y_2, Z_2 \in \mathbb{G}_{p_2}$ at random, and output $C_w = (C_1 = g^r Y_2,\ C_2 = H(w)^r Z_2)$.
- $\mathsf{Test}(PK, C_w, T_w)$: Using the notation in the description of Td and PEKS, if $\mathbf{e}(T_1, C_2) = \mathbf{e}(T_2, C_1)$, then output 1; otherwise, output 0.

From the orthogonality property, the completeness and consistency of Π_3 can readily be verified. To show the security of Π_3, we introduce the following assumptions.

Assumption II We say that $\mathcal{G}_3$ *satisfies Assumption II* if for any PPT algorithm $\mathcal{B}$,

$$\mathsf{Adv}^{\text{A-2}}_{\mathcal{A},\mathcal{G}_3}(k) = \Big| \Pr\Big[I \leftarrow \mathcal{G}_3(1^k) \ ; \ N \leftarrow p_1p_2p_3 \ ; \ g_i \leftarrow \mathbb{G}^*_{p_i} \quad (1 \le i \le 3) \ ; \\ X_0, X_1 \leftarrow \mathbb{G}_{p_1} \ ; \ s, s' \leftarrow \mathbb{Z}_N \ ; \ R_3, S_3, R'_3, S'_3 \leftarrow \mathbb{G}_{p_3} \ ; \ b \leftarrow \{0,1\} \ ; \\ b' \leftarrow \mathcal{B}\left(1^k, N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_1, g_2, g_3, (g_1^s R_3, X_0^s S_3), (g_1^{s'} R'_3, X_b^{s'} S'_3)\right) \ : \ b = b' \Big] - \frac{1}{2} \Big|$$

is negligible.

Assumption III We say that $\mathcal{G}_3$ *satisfies Assumption III* if for any PPT algorithm $\mathcal{B}$,

$$\mathsf{Adv}^{\text{A-3}}_{\mathcal{A},\mathcal{G}_3}(k) = \Big| \Pr\Big[I \leftarrow \mathcal{G}_3(1^k) \ ; \ N \leftarrow p_1p_2p_3 \ ; \ g_i \leftarrow \mathbb{G}^*_{p_i} \ (1 \le i \le 3) \ ; \ \alpha, \beta \leftarrow \mathbb{Z}^*_N \ ; \\ X_0, X_1 \leftarrow \mathbb{G}_{p_1} \ ; \ r \leftarrow \mathbb{Z}_N \ ; \ R_2, Y_2, Z_2 \leftarrow \mathbb{G}_{p_2} \ ; \ g \leftarrow g_1 R_2 \ ; \ b \leftarrow \{0,1\} \ ; \\ b' \leftarrow \mathcal{B}\left(1^k, N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_2, g_3, g_1^\alpha, g_1^\beta, g_1^{\alpha\beta}, g, X_0, X_1, (g^r Y_2, X_b^r Z_2)\right) : b = b' \Big] - \frac{1}{2} \Big|$$

is negligible.

Assumption IV We say that $\mathcal{G}_3$ *satisfies Assumption IV* if for any PPT algorithm $\mathcal{B}$,

$$\mathsf{Adv}^{\text{A-4}}_{\mathcal{A},\mathcal{G}_3}(k) = \Big| \Pr\Big[I \leftarrow \mathcal{G}_3(1^k) \ ; \ N \leftarrow p_1p_2p_3 \ ; \ h_0, h_1 \leftarrow (\mathbb{G}_T)_{p_1} \ ; \ \alpha, \beta \leftarrow \mathbb{Z}^*_N \ ; \\ b \leftarrow \{0,1\} \ ; \ b' \leftarrow \mathcal{B}\left(1^k, N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, h_0, h_1, h_b^\alpha, h_b^\beta, h_b^{\alpha\beta}\right) \ : \ b = b' \Big] - \frac{1}{2} \Big|$$

is negligible.

It can readily be shown that if $\mathcal{G}_3$ satisfies Assumption II then it also satisfies the DDH assumption over $(\mathbb{G}_T)_{p_1}$. Thus, Assumption II is a stronger assumption than the DDH assumption over $(\mathbb{G}_T)_{p_1}$. Assumption IV is presented to explain the position of Assumption III but with a simpler representation. Proposition 3 says that Assumption III is a stronger assumption than Assumption IV its proof is straightforward and left to the reader.

Proposition 3. *If $\mathcal{G}_3$ satisfies Assumption III, then it also satisfies Assumption IV.*

Proposition 4. *Suppose that H is an RO. For any probability ensemble $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ and any SPP adversary $\mathcal{A}$ that makes at most $q_H(k)$ queries to H and at most $q_t(k)$ trapdoor queries when the security parameter k is given, there exists a PPT algorithm $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\text{spp}}_{\mathcal{A},\Pi_3,\mathcal{X}}(k) \le \mathsf{Adv}^{\text{A-2}}_{\mathcal{B},\mathcal{G}_2}(k) + 2(q_H(k) + q_t(k)) \cdot ||\mathbf{X}_k|| \quad (k \in \mathbb{N}).$$

Proposition 5. *Suppose that H is an RO. For any probability ensemble $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ and any IND-PKP adversary $\mathcal{A}$ against Π_3 that makes at most $q_H(k)$ queries*[3] *to H and at most $q_t(k)$ trapdoor queries when the security parameter k is given,*

$$\mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A},\Pi_3,\mathcal{X}}(k) \leq 2(q_H(k) + q_t(k)) \cdot ||\mathbf{X}_k|| \quad (k \in \mathbb{N}).$$

Proposition 6. *Suppose that H is an RO. For any IND-PEKS-CKA adversary $\mathcal{A}$ against Π_3 that makes at most $q_H(k)$ queries to H when the security parameter k is given, there exists a PPT algorithm $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\text{ind:peks}}_{\mathcal{A},\Pi_3}(k) \leq (q_H(k)+1)(q_H(k)+2) \cdot \mathsf{Adv}^{\text{A-3}}_{\mathcal{B},\mathcal{G}_3}(k) \quad (k \in \mathbb{N}).$$

The proofs of Propositions 4, 5, and 6 are given in Appendixes D, E, and F respectively. The open problem is to construct a PEKS scheme that is IND-PKP+ secure and IND-PEKS-CKA secure, either in the standard model or the RO model, under reasonable assumptions.

Remark 7. We now consider a PEKS scheme that is identical with Π_3 except that the searchable ciphertext of w is given by $C_w = (C_1 = g^r,\ C_2 = H(w)^r)$. From a similar discussion to that for Proposition 5, it can be shown that this PEKS scheme is IND-PKP secure; however, it is not IND-PEKS-CKA secure. This is because for a given target ciphertext $C_{w_b} = (C_1, C_2)$, an adversary can easily guess the challenge bit b by outputting $b' \in \{0,1\}$ such that $\mathbf{e}(g, C_2) = \mathbf{e}(C_1, H(w_{b'}))$. This instance demonstrates the separation between the IND-PKP and IND-PEKS-CKA securities.

5 Postscript

We have introduced new security notions for PEKS systems, namely PKP, IND-PKP, PKP+, and IND-PKP+, which take account of the privacy of a keyword from a trapdoor. We have also showed that these notions ensure strictly weaker security with respect to keyword leakage from only the ciphertext, as compared to IND-PEKS-CKA security. Accordingly, for achieving higher security in PEKS, we have presented several instances of a PEKS scheme that is IND-PKP or IND-PKP+ secure, in addition to being IND-PEKS-CKA secure. From a practical viewpoint, however, we have no corroboration that either IND-PKP or IND-PKP+ security is insufficient to ensure the privacy of a keyword from a ciphertext. We expect that the underlying notion and PRIV security [3] give equal security levels, because they are defined for the situation in which the target keywords are chosen from a well-spread distribution, and the (guessing) adversary cannot see them. We are sure that it is easier to design efficient IND-PKP (or IND-PKP+) secure PEKS schemes than it is to design efficient IND-PEKS-CKA

[3] In the PEKS scheme Π_3, H cannot be used as a common parameter for all users because Π_3 depends on a composite bilinear map. Hence, an IND-PKP adversary can make queries to both H and H'. For simplicity, we assume that the total numbers of queries to both H and H' is written by $q_H(k)$.

secure PEKS schemes. Indeed, we can use secure injective-function generators to achieve IND-PKP security, and it is easy to design practical injective-function generators that are secure under reasonable assumptions.

References

1. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, 2008.
2. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in multi-user setting: Security proofs and improvements. In *Advances in Cryptology – Eurocrypt 2000*, LNCS 1807, pages 259–274. Springer-Verlag, 2000.
3. M. Bellare, A. Boldyreva, and A. O'Neil. Deterministic and efficiently searchable encryption. In *Advances in Cryptology – Crypto 2007*, LNCS 4622, pages 535–552. Springer-Verlag, 2007.
4. C. Blundo, V. Iovino, and G. Persiano. Predicate encryption with partial public keys. In *Cryptology And Network Security (CANS 2010)*, LNCS 6467, pages 298–313. Springer-Verlag, 2010.
5. D. Boneh, Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Advances in Cryptology – Eurocrypt 2004*, LNCS 3027, pages 506–522. Springer-Verlag, 2004.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology – Crypto 2001*, LNCS 2139, pages 213–229. Springer-Verlag, 2001.
7. D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography – TCC 2011*, LNCS 6597, pages 253–273. Springer-Verlag, 2011.
8. X. Boyen and B. Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *Advances in Cryptology – Crypto 2006*, LNCS 4117, pages 290–307. Springer-Verlag, 2006.
9. J. Camenisch, M. Kohlweiss, A. Rial, and C. Sheedy. Blind and anonymous identity-based encryption and authorized private searches on public-key encrypted data. In *Public Key Cryptography 2009*, LNCS 5443, pages 196–214. Springer-Verlag, 2009.
10. R. Canetti. Towards realizing random oracles: Hash functions that hide all partial information. In *Advances in Cryptology – Crypto '97*, LNCS 1294, pages 455–469. Springer-Verlag, 1997.
11. R. Canetti, D. Micciancio, and O. Reingold. Perfectly one-way probabilistic hash functions. Proceedings of the 30th ACM STOC '98, pages 131–140, 1998.
12. A. De Caro, V. Iovino, and G. Persiano. Hidden vector encryption fully secure against unrestricted queries. IACR Cryptology ePrint Archive, Report 2011/546. Available from `http://eprint.iacr.org/2011/546`.
13. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Confeence on Cryptography and Coding*, pages 26–28, 2001.
14. R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *Proceedings of the 13th ACM Conference on Computer and Communication Security*, pages 79–88, 2006.

15. C. Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology – Eurocrypt 2006*, LNCS 4004, pages 445–464. Springer-Verlag, 2006.
16. A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Advances in Cryptology – Eurocrypt 2005*, LNCS 3493, pages 457–473. Springer-Verlag, 2005.
17. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Crypto '84*, LNCS 196, pages 47–53. Springer-Verlag, 1984.
18. E. Shen, E. Shi, and B. Waters. Predicate privacy in encryption systems. In *Theory of Cryptography – TCC 2009*, LNCS 5444, pages 457–473. Springer-Verlag, 2009.
19. B. Waters, D. Balfanz, G. Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *NDSS*. The Internet Society, 2004.

A Proof of Theorem 1

Let $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ be a well-spread probability ensemble. We show that if the PEKS scheme Π is IND-PKP secure with respect to $\mathcal{X}$, then it has PKP with respect to $\mathcal{X}$. For an IND-PKP adversary $\mathcal{A}$, we define

$$\rho^{(1)}_{\mathcal{A},\Pi,\mathcal{X}}(k) = \Pr\Big[(PK,SK),(PK',SK') \leftarrow \mathsf{KG}(1^k)\ ;\ w \leftarrow \mathbf{X}_k\ ;\ T_w \leftarrow \mathsf{Td}(SK,w)\ ;\ C_w \leftarrow \mathsf{PEKS}(PK,w)\ ;\ T'_w \leftarrow \mathsf{Td}(SK',w)\ ;\ C'_w \leftarrow \mathsf{PEKS}(PK',w)\ :\ \mathcal{A}^{\mathsf{Td}(SK,\cdot),\mathsf{Td}(SK',\cdot)}(1^k,PK,T_w,C_w,PK',T'_w,C'_w) = 1\Big],$$

$$\rho^{(2)}_{\mathcal{A},\Pi,\mathcal{X}}(k) = \Pr\Big[(PK,SK),(PK',SK') \leftarrow \mathsf{KG}(1^k)\ ;\ w,w' \leftarrow \mathbf{X}_k\ ;\ T_w \leftarrow \mathsf{Td}(SK,w)\ ;\ C_w \leftarrow \mathsf{PEKS}(PK,w)\ ;\ T'_{w'} \leftarrow \mathsf{Td}(SK',w')\ ;\ C'_{w'} \leftarrow \mathsf{PEKS}(PK',w') : \mathcal{A}^{\mathsf{Td}(SK,\cdot),\mathsf{Td}(SK',\cdot)}(1^k,PK,T_w,C_w,PK',T'_{w'},C'_{w'}) = 1\Big].$$

Then we have

$$2 \cdot \mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A},\Pi,\mathcal{X}}(k) = \left|\rho^{(1)}_{\mathcal{A},\Pi,\mathcal{X}}(k) - \rho^{(2)}_{\mathcal{A},\Pi,\mathcal{X}}(k)\right|. \tag{2}$$

We now suppose that Π does not have PKP with respect to $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$. Then from Definition 3, there exists a predicate family $\mathcal{P} = \{P_k\}_{k\in\mathbb{N}}$ and a PPT algorithm $\mathcal{B}$ such that for any PPT algorithm $\mathcal{C}$,

$$\rho(k) = \Pr\left[\mathsf{Exp}^{\text{pkp:real}}_{\mathcal{B},\Pi,\mathcal{X},\mathcal{P},t}(k) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{pkp:ideal}}_{\mathcal{C},\Pi,\mathcal{X},\mathcal{P}}(k) = 1\right] \tag{3}$$

is non-negligible. We now consider a PPT algorithm $\mathcal{C}$ (in the ideal system) that works as follows:

1. Select a random sample $w' \leftarrow \mathbf{X}_k$ and make a trapdoor query w' to obtain the trapdoor $T_{w'}$. Generate a searchable ciphertext of w' by $C_{w'} \leftarrow \mathsf{PEKS}(PK,w')$.
2. Run $\mathcal{B}$ on input $(1^k, PK, T_{w'}, C_{w'})$, and output the corresponding response of $\mathcal{B}$. If $\mathcal{B}$ makes trapdoor queries then respond to them by using $\mathcal{C}$'s trapdoor oracle.

This completes the description of $\mathcal{C}$. Moreover, for each $w \in [\mathbf{X}_k]$, we define

$$\zeta_{w;\mathcal{B}}(k) = \Pr\Big[(PK, SK) \leftarrow \mathsf{KG}(1^k)\ ;\ T_w \leftarrow \mathsf{Td}(SK, w)\ ;\\ C_w \leftarrow \mathsf{PEKS}(PK, w)\ :\ \mathcal{B}^{\mathsf{Td}(SK,\cdot)}(1^k, PK, T_w, C_w) = 1\Big].$$

Let $S^i_{P_k}$ denote a set $\{w \in [\mathbf{X}_k] \mid P_k(w) = i\}$, for each $i \in \{0, 1\}$. Then, since $\mathbf{X}_k$ is independent from the key generation, we have

$$\Pr\left[\mathsf{Exp}^{\text{pc:real}}_{\mathcal{B},\Pi,\mathcal{X},\mathcal{P}}(k) = 1\right] = \sum_{w \in S^1_{P_k}} \Pr[\mathbf{X}_k = w] \cdot \zeta_{w;\mathcal{B}}(k) \\ + \sum_{w \in S^0_{P_k}} \Pr[\mathbf{X}_k = w] \cdot (1 - \zeta_{w;\mathcal{B}}(k)), \tag{4}$$

$$\Pr\left[\mathsf{Exp}^{\text{pc:ideal}}_{\mathcal{B},\Pi,\mathcal{X},\mathcal{P}}(k) = 1\right] = \sum_{w \in S^1_{P_k}} \Pr[\mathbf{X}_k = w] \cdot \sum_{w' \in [\mathbf{X}_k]} \Pr[\mathbf{X}_k = w'] \cdot \zeta_{w';\mathcal{B}}(k) \\ + \sum_{w \in S^0_{P_k}} \Pr[\mathbf{X}_k = w] \cdot \sum_{w' \in [\mathbf{X}_k]} \Pr[\mathbf{X}_k = w'] \cdot (1 - \zeta_{w';\mathcal{B}}(k)). \tag{5}$$

Let $\mathbf{Z}_k$ be a random variable over $[0, 1] = \{a \in \mathbb{R} \mid 0 \le a \le 1\}$ such that $\Pr[\mathbf{Z}_k = \zeta_{w;\mathcal{B}}(k)] = \Pr[\mathbf{X}_k = w]$. Then, from (3), (4), and (5), we have

$$\begin{aligned}
\rho(k) &= \sum_{w \in S^1_{P_k}} \Pr[\mathbf{X}_k = w] \cdot \left(\zeta_{w;\mathcal{B}}(k) - \sum_{w' \in [\mathbf{X}_k]} \Pr[\mathbf{X}_k = w'] \cdot \zeta_{w';\mathcal{B}}(k)\right) \\
&\quad - \sum_{w \in S^0_{P_k}} \Pr[\mathbf{X}_k = w] \cdot \left(\zeta_{w;\mathcal{B}}(k) - \sum_{w' \in [\mathbf{X}_k]} \Pr[\mathbf{X}_k = w'] \cdot \zeta_{w',\mathcal{B}}(k)\right) \\
&\le \sum_{w \in [\mathbf{X}_k]} \Pr[\mathbf{X}_k = w] \cdot \left|\zeta_{w;\mathcal{B}}(k) - \sum_{w' \in [\mathbf{X}_k]} \Pr[\mathbf{X}_k = w'] \cdot \zeta_{w',\mathcal{B}}(k)\right| \\
&= E(|\mathbf{Z}_k - E(\mathbf{Z}_k)|) \le \sqrt{E(\mathbf{Z}_k^2) - E(\mathbf{Z}_k)^2}.
\end{aligned} \tag{6}$$

Let $\mathcal{A}^*$ be an IND-PKP adversary such that for a given input $(1^k, PK, T_w, C_w, PK', \bar{T}, \bar{C})$, where $(\bar{T}, \bar{C})$ is (T'_w, C'_w) or $(T'_{w'}, C'_{w'})$, it runs $\mathcal{B}$ twice and outputs 1 only when $\mathcal{B}(1^k, PK, T_w, C_w) = \mathcal{B}(1^k, PK', \bar{T}, \bar{C}) = 1$. Then we have

$$E(\mathbf{Z}_k^2) = \rho^{(1)}_{\mathcal{A}^*,\Pi,\mathcal{X}}(k) \qquad \text{and} \qquad E(\mathbf{Z}_k)^2 = \rho^{(2)}_{\mathcal{A}^*,\Pi,\mathcal{X}}(k). \tag{7}$$

From (2), (6), and (7), we finally have

$$\rho(k)^2 \le \left|\rho^{(1)}_{\mathcal{A}^*,\Pi,\mathcal{X}}(k) - \rho^{(2)}_{\mathcal{A}^*,\Pi,\mathcal{X}}(k)\right| = \frac{1}{2}\mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A}^*,\Pi,\mathcal{X}}(k).$$

This contradicts the assumption that Π is IND-PKP secure.

B Proof of Theorem 2

We think that this theorem is almost trivial. Therefore, we describe its proof only briefly. We first show (a). Suppose that there exists an IND-PKP adversary $\mathcal{A}$. We then construct a PPT algorithm $\mathcal{B}$ for breaking the security of $(\mathcal{I}, \mathfrak{G})$. $\mathcal{B}$ takes $(\lambda_k, \pi, \pi', A = \pi(x_0), B = \pi'(x_b))$ as input, and its goal is to guess $b \in \{0,1\}$.

1. Run KG to obtain $(PK, SK), (PK', SK') \leftarrow \mathsf{KG}(1^k)$, and set $(PK^*, SK^*) = ((PK, \lambda_k, \pi), (SK, \lambda_k, \pi))$ and $(PK'^*, SK'^*) = ((PK', \lambda_k, \pi'), (SK', \lambda_k, \pi'))$.
2. Set $T = \mathsf{Td}(SK, A)$, $T' = \mathsf{Td}(SK, B)$, $C = \mathsf{PEKS}(PK, A)$, and $C' = \mathsf{PEKS}(PK', B)$, and run $\mathcal{A}$ on input $(1^k, PK^*, T, C, PK'^*, T', C')$.
3. If $\mathcal{A}$ makes a trapdoor query x to $\mathsf{Td}^*(\hat{SK}, \cdot)$, where $\hat{SK} \in \{SK^*, SK'^*\}$, then return the result of running $\mathsf{Td}^*(\hat{SK}, x)$.
4. If $\mathcal{A}$ finally outputs $b' \in \{0,1\}$, then output it as its guess for b.

This completes the description of $\mathcal{B}$. From the specification of $\mathcal{B}$, it is clear that $\mathcal{B}$ completely simulates the IND-PKP experiment of Π^*. Therefore, we have $\mathsf{Adv}^{\mathrm{sif}}_{\mathcal{B},\mathcal{I},\mathfrak{G},\mathcal{X}}(k) = \mathsf{Adv}^{\text{ind-pkp}}_{\mathcal{A},\Pi^*,\mathcal{X}}(k)$.

Next, we show (b). Suppose that there exists an IND-PEKS-CKA adversary $\mathcal{A}$ against Π^*. We then construct an IND-PEKS-CKA adversary $\mathcal{B}$ against Π. $\mathcal{B}$ takes $(1^k, PK)$ as input, and works as follows.

1. Run $\mathcal{I}$ and $\mathfrak{G}$ to obtain $\lambda_k \leftarrow \mathcal{I}(1^k)$ and $\pi \leftarrow \mathfrak{G}(\lambda_k)$, and run $\mathcal{A}$ on input $(1^k, PK^* = (PK, \lambda_k, \pi))$.
2. If $\mathcal{A}$ makes a trapdoor query x, then make a query $\pi(x)$ to $\mathcal{B}$'s trapdoor oracle and respond with the corresponding response to $\mathcal{A}$.
3. If $\mathcal{A}$ makes a challenge query (x_0, x_1), then make a query $(\pi(x_0), \pi(x_1))$ to $\mathcal{B}$'s challenge oracle and respond with the corrsponding response to $\mathcal{A}$. Note that $\pi(x_0) \neq \pi(x_1)$ because $x_0 \neq x_1$ and π is injective.
4. If $\mathcal{A}$ finally outputs $b' \in \{0,1\}$, then output it and halt.

This completes the description of $\mathcal{B}$. From the specification of $\mathcal{B}$, it is clear that $\mathcal{B}$ completely simulates the IND-PEKS-CKA experiment of Π^*. Therefore, we have $\mathsf{Adv}^{\text{ind:peks}}_{\mathcal{B},\Pi}(k) = \mathsf{Adv}^{\text{ind:peks}}_{\mathcal{A},\Pi^*}(k)$.

C Proof of Proposition 2

For a PPT algorithm $\mathcal{B}$, we now consider the following experiment.

Experiment $\mathsf{Exp}_{\mathcal{B},\mathcal{G}}(k)$
$(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}(1^k)$; $g \leftarrow \mathbb{G}^*$; $a, a' \leftarrow \mathbb{Z}_p$; $v_0, v'_1 \leftarrow \mathbb{G}$
If $a = a'$ then $v'_0 = v_0$ else $v'_0 \leftarrow \mathbb{G}$
$b \leftarrow \{0,1\}$; $b' \leftarrow \mathcal{B}(1^k, p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, g^a, g^{a'}, v_0^a, {v'_b}^{a'})$
If $b = b'$ then return 1 else return 0.

Then it is trivial that for any PPT algorithm $\mathcal{B}$,

$$\mathsf{Adv}_{\mathcal{B},\mathcal{G}}(k) = \left|\Pr[\mathsf{Exp}_{\mathcal{B},\mathcal{G}}(k)=1] - \frac{1}{2}\right| \leq \frac{1}{p} < \frac{1}{2^k}. \tag{8}$$

By using this result, we show that Π_2 is IND-PKP secure. Let $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ be a well-spread probability ensemble. We suppose that there exists an IND-PKP adversary $\mathcal{A}$ against Π_2 with respect to $\mathcal{X}$. Then we build a PPT algorithm $\mathcal{B}$ in the experiment $\mathsf{Exp}_{\mathcal{B},\mathcal{G}}(k)$, which is based on $\mathcal{A}$. $\mathcal{B}$ takes $(1^k, p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h = g^a, h' = g^{a'}, Y = v_0^a, Z = {v'_b}^{a'})$ as input, and works as follows. The goal of $\mathcal{B}$ is to guess the challenge bit b.

1. Select random samples $x_0, x_1 \leftarrow \mathbf{X}_k$.
2. **Public key simulation.** Set $PK = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h)$ and $PK' = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g, h')$. Note that $SK = (PK, a)$ and $SK' = (PK', a')$ in this case.
3. **Trapdoor simulation.** Set $T_{x_0} = Y$ and $T'_{x_b} = Z$. Note that here $\mathcal{B}$ implicitly sets $H_1(PK||x_0) = v_0$ and $H_1(PK'||x_b) = v'_b$. See Step 6 concerning the simulation of H_1.
4. **Ciphertext simulation.** Pick $r, r' \in \mathbb{Z}_p$ at random, and set $C = (g^r, H_2(\mathbf{e}(Y, g^r)))$ and $C' = (g^{r'}, H_2(\mathbf{e}(Z, g^{r'})))$. See Step 6 for how to compute $H_2(\cdot)$. Note that since

$$\mathbf{e}(Y, g^r) = \mathbf{e}(v_0^a, g^r) = \mathbf{e}(H_1(PK||x_0), h^r) \quad \text{and}$$
$$\mathbf{e}(Z, g^{r'}) = \mathbf{e}({v'_b}^{a'}, g^{r'}) = \mathbf{e}(H_1(PK'||x_b), {h'}^{r'}),$$

   C and C' are valid random ciphertexts of x_0 and x_b under PK and PK', respectively.
5. Run $\mathcal{A}$ on input $(1^k, PK, T_{x_0}, C, PK', T'_{x_b}, C')$.
6. **Random oracle queries.** To respond to the RO queries from $\mathcal{A}$, $\mathcal{B}$ maintains lists L_{H_1} and L_{H_2}, which are initially empty. Suppose that $\mathcal{A}$ makes an H_1-query $y = \hat{PK}||x$, where $\hat{PK} \in \{PK, PK'\}$.
   (a) If $x \in \{x_0, x_1\}$, then output a failure message and halt; otherwise, do the following.
   (b) If $(y, r_y, H_1(y)) \in L_{H_1}$ for some $(r_y, H_1(y)) \in \mathbb{Z}_p \times \mathbb{G}$, then respond with $H_1(y)$ to $\mathcal{A}$. Otherwise, pick a random $r_y \in \mathbb{Z}_p$, add $(y, r_y, H_1(y) = g^{r_y})$ to L_{H_1}, and respond with $H_1(y)$ to $\mathcal{A}$.

   Suppose that $\mathcal{A}$ makes an H_2-query $z \in \mathbb{G}_T$. If $(z, H_2(z)) \in L_{H_2}$ for some $H_2(z)$, then respond with $H_2(z)$ to $\mathcal{A}$. Otherwise, pick a random $H_2(z)$ from the range of H_2, add $(z, H_2(z))$ to L_{H_2}, and respond with $H_2(z)$ to $\mathcal{A}$.
7. **Trapdoor queries.** Suppose that $\mathcal{A}$ makes a query x to $\mathsf{Td}(\hat{SK}, \cdot)$, where $\hat{SK} \in \{SK, SK'\}$.
   (a) If $x \in \{x_0, x_1\}$, then output a failure message and halt; otherwise, do the following.
   (b) Seek a tuple $(y = \hat{PK}||x, r_y, H_1(y)) \in L_{H_1}$; if no such tuple exists, do the same procedure as in Step 6(b) to obtain it. Respond to $\mathcal{A}$ with $T_x = h^{r_y}$

if $\hat{SK} = SK$; otherwise (i.e., if $\hat{SK} = SK'$), respond with $T'_x = {h'}^{r_y}$ to $\mathcal{A}$. Note that since

$$T_x = h^{r_y} = g^{ar_y} = H_1(PK||x)^a \quad \text{and} \quad T'_x = {h'}^{r_y} = g^{a'r_y} = H_1(PK'||x)^{a'},$$

T_x and T'_x are valid random trapdoors for the keyword x under SK and SK', respectively.

8. **Guess.** If $\mathcal{A}$ finally outputs $b' \in \{0, 1\}$, then output b' as its guess for b.

This completes the description of $\mathcal{B}$. For the running time of $\mathcal{B}$, we have $T_{\mathcal{B}}(k) = T_{\mathcal{A}}(k) + O(k^c)$ for some constant $c > 0$. For simplicity, let $\mathsf{Exp}_0 = \mathsf{Exp}^{\text{ind-pkp}}_{\mathcal{A},\Pi_2,\mathcal{X}}(k)$. To analyze the advantage of $\mathcal{B}$, we define the following events in Exp_0.

- E_{H_1} is an event in which $\mathcal{A}$ makes an H_1-query $\hat{PK}||x$ such that $\hat{PK} \in \{PK, PK'\}$ and $x \in \{x_0, x_1\}$.
- E_t is an event in which $\mathcal{A}$ makes a trapdoor query x such that $x \in \{x_0, x_1\}$.
- $\mathsf{E} = \mathsf{E}_{H_1} \vee \mathsf{E}_t$.

Let Exp_1 be an experiment that is identical with Exp_0 except that if E occurs, it outputs a failure message. From the specification of $\mathcal{B}$, it completely simulates Exp_1. Hence, we have

$$\mathsf{Adv}_{\mathcal{B},\mathcal{G}}(k) = \left|\Pr[\mathsf{Exp}_1 = 1] - \frac{1}{2}\right|. \tag{9}$$

Without loss of generality, we can assume that $\Pr[\mathsf{Exp}_0 = 1] - \frac{1}{2} \geq 0$. From the definition of Exp_1, $\Pr[\mathsf{Exp}_0 = 1 \mid \neg\mathsf{E}] = \Pr[\mathsf{Exp}_1 = 1 \mid \neg\mathsf{E}]$ and $\Pr[\mathsf{Exp}_1 = 1 \mid \mathsf{E}] = 0$. Therefore, we have

$$\begin{aligned}\Pr[\mathsf{Exp}_0 = 1] - \frac{1}{2} &= \Pr[\mathsf{Exp}_0 = 1 \mid \neg\mathsf{E}] \cdot \Pr[\neg\mathsf{E}] + \Pr[\mathsf{Exp}_0 = 1 \mid \mathsf{E}] \cdot \Pr[\mathsf{E}] - \frac{1}{2} \\ &\leq \Pr[\mathsf{Exp}_1 = 1 \mid \neg\mathsf{E}] \cdot \Pr[\neg\mathsf{E}] - \frac{1}{2} + \Pr[\mathsf{E}] = \Pr[\mathsf{Exp}_1 = 1] - \frac{1}{2} + \Pr[\mathsf{E}].\end{aligned} \tag{10}$$

Next, we examine $\Pr[\mathsf{E}]$. Since H_1 is an RO, there is no efficient way to guess x_0 or x_1 beyond random sampling from $\mathbf{X}_k$. Hence, we have

$$\begin{aligned}\Pr[\mathsf{E}_{H_1}] \leq 1 - (1 - 2||\mathbf{X}_k||)^{q_{H_1}(k)} \leq 2q_{H_1}(k) \cdot ||\mathbf{X}_k|| \qquad \text{and} \\ \Pr[\mathsf{E}_t \mid \neg\mathsf{E}_{H_1}] \leq 1 - (1 - 2||\mathbf{X}_k||)^{q_t(k)} \leq 2q_t(k) \cdot ||\mathbf{X}_k||.\end{aligned}$$

It follows that

$$\begin{aligned}\Pr[\mathsf{E}] &= \Pr[\mathsf{E}_t \mid \mathsf{E}_{H_1}] \cdot \Pr[\mathsf{E}_{H_1}] + \Pr[\mathsf{E}_t \mid \neg\mathsf{E}_{H_1}] \cdot \Pr[\neg\mathsf{E}_{H_1}] \\ &\leq \Pr[\mathsf{E}_{H_1}] + \Pr[\mathsf{E}_t \mid \neg\mathsf{E}_{H_1}] \leq 2(q_{H_1}(k) + q_t(k)) \cdot ||\mathbf{X}_k||.\end{aligned} \tag{11}$$

From (8), (9), (10) and (11), we can obtain the claimed result.

D Proof of Proposition 4

For given probability ensemble $\mathcal{X} = \{\mathbf{X}_k\}_{k\in\mathbb{N}}$ and SPP adversary $\mathcal{A}$ against Π_3, we construct an algorithm $\mathcal{B}$ that has the underlying feature. $\mathcal{B}$ takes $(1^k, N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_1, g_2, g_3, T = (g_1^s R_3, X_0^s S_3), T' = (g_1^{s'} R'_3, X_b^{s'} S'_3))$ as input, and its goal is to guess $b \in \{0, 1\}$.

1. Pick random samples $x_0, x_1 \leftarrow \mathbf{X}_k$ and a random $R_2 \in \mathbb{G}_{p_2}$, and set $PK = (1^k, N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_2, g_3, g = g_1 R_2)$.
2. Run $\mathcal{A}$ on input $(1^k, PK, T, T')$. Note that here $\mathcal{B}$ implicitly set $H(x_0) = X_0$ and $H(x_1) = X_1$ (although $\mathcal{B}$ cannot know those). Obviously, T and T' have the same distributions as the (valid) trapdoors of x_0 and x_b respectively.
3. **RO queries.** To respond to $\mathcal{A}$'s queries to H, $\mathcal{B}$ maintains a list L_H which is initially empty. Suppose that $\mathcal{A}$ makes an H-query x.
   (a) If $x \in \{x_0, x_1\}$, then output a failure message and halt; otherwise, do the following steps.
   (b) If $(x, H(x)) \in L_H$ for some $H(x)$, then respond with $H(x)$ to $\mathcal{A}$. Otherwise, pick a random $r \in \mathbb{Z}_n$, add $(x, H(x) = g_1^r)$ to L_H, and respond with $H(x)$ to $\mathcal{A}$.
4. **Trapdoor queries.** Suppose that $\mathcal{A}$ makes a trapdoor query x.
   (a) If $x \in \{x_0, x_1\}$, then output a failure message and halt; otherwise, do the following steps.
   (b) Seek a tuple $(x, H(x)) \in L_H$; if no such tuple exists, do the same procedure as in Step 3(b) to obtain it. Pick $s \in \mathbb{Z}_n$ and $R_3, S_3 \in \mathbb{G}_{p_3}$ at random, and return $T_x = (g_1^s R_3, H_1(x)^s S_3)$ to $\mathcal{A}$.
5. **Guess.** If $\mathcal{A}$ finally outputs $b' \in \{0, 1\}$, output it as its guess for b.

This completes the description of $\mathcal{B}$. For the running time of $\mathcal{B}$, we have $T_{\mathcal{B}}(k) = T_{\mathcal{A}}(k) + O(k^c)$ for some constant $c > 0$. For simplicity, let $\mathsf{Exp}_0 = \mathsf{Exp}^{\mathrm{spp}}_{\mathcal{A},\Pi_3,\mathcal{X}}(k)$. To analyze the advantage of $\mathcal{B}$, we define the following events in Exp_0.

- E_H is an event in which $\mathcal{A}$ makes an H-query $x \in \{x_0, x_1\}$.
- E_t is an event in which $\mathcal{A}$ makes a trapdoor query $x \in \{x_0, x_1\}$.
- $\mathsf{E} = \mathsf{E}_H \vee \mathsf{E}_t$.

Let Exp_1 be an experiment that is identical with Exp_0 except that if E occurs, it outputs a failure message. From the specification of $\mathcal{B}$, it completely simulates Exp_1. Hence, we have

$$\mathsf{Adv}^{\text{A-1}}_{\mathcal{B},\mathcal{G}_3}(k) = \left|\Pr[\mathsf{Exp}_1 = 1] - \frac{1}{2}\right|. \tag{12}$$

Without loss of generality, we can assume $\Pr[\mathsf{Exp}_0 = 1] - 1/2 > 0$. From the definition of Exp_1, $\Pr[\mathsf{Exp}_0 = 1 \mid \neg\mathsf{E}] = \Pr[\mathsf{Exp}_1 = 1 \mid \neg\mathsf{E}]$ and $\Pr[\mathsf{Exp}_1 = 1 \mid \mathsf{E}] = 0$. Therefore, we have

$$\Pr[\mathsf{Exp}_0 = 1] - \frac{1}{2} = \Pr[\mathsf{Exp}_0 = 1 \mid \neg\mathsf{E}] \cdot \Pr[\neg\mathsf{E}] + \Pr[\mathsf{Exp}_0 = 1 \mid \mathsf{E}] \cdot \Pr[\mathsf{E}] - \frac{1}{2}$$

$$\leq \Pr[\mathsf{Exp}_1 = 1 \mid \neg\mathsf{E}] \cdot \Pr[\neg\mathsf{E}] - \frac{1}{2} + \Pr[\mathsf{E}] \leq \Pr[\mathsf{Exp}_1 = 1] - \frac{1}{2} + \Pr[\mathsf{E}]. \tag{13}$$

On the other hand, from a discussion similar to that for deriving (11), we have

$$\Pr[\mathsf{E}] \leq 2(q_H(k) + q_t(k)) \cdot ||\mathbf{X}_k||. \tag{14}$$

From (12), (13), and (14), we can obtain the claimed result.

E Proof of Proposition 5

For a PPT algorithm $\mathcal{B}$, we now consider the following experiment.

Experiment $\mathsf{Exp}_{\mathcal{B},\mathcal{G}_3}(k)$
$(p'_1, p'_2, p'_3, \mathbb{G}', \mathbb{G}'_T, \mathbf{e}') \leftarrow \mathcal{G}_3(1^k)$; $X_0, X_1 \leftarrow \mathbb{G}'_{p'_1}$
$b \leftarrow \{0,1\}$; $b' \leftarrow \mathcal{B}(1^k, p'_1, p'_2, p'_3, \mathbb{G}', \mathbb{G}'_T, \mathbf{e}', X_b)$
If $b = b'$ then return 1 else return 0.

It is then obvious that $\Pr[\mathsf{Exp}_{\mathcal{B},\mathcal{G}_3}(k) = 1] = 1/2$ for any $\mathcal{B}$. For given IND-PKP adversary $\mathcal{A}$ against Π_3, we construct an algorithm $\mathcal{B}$ in the above experiment in order to evaluate the advantage of $\mathcal{A}$. $\mathcal{B}$ takes $(1^k, p'_1, p'_2, p'_3, \mathbb{G}', \mathbb{G}'_T, \mathbf{e}', X_b)$ as input, and its goal is to guess $b \in \{0,1\}$.

1. Pick random samples $(p_1, p_2, p_3, \mathbb{G}, \mathbb{G}_T, \mathbf{e}) \leftarrow \mathcal{G}_3(1^k)$ and $x_0, x_1 \leftarrow \mathbf{X}_k$.
2. Pick a random $A_0 \in \mathbb{G}_{p_1}$, and add (x_0, A_0) to the list L_H, which is initially empty. Thus, we set $H(x_0) = A_0$ in the simulation of H (cf. Step 7).
3. Pick $g_i \in \mathbb{G}^*_{p_i}$, $g'_i \in \mathbb{G}'^*_{p'_i}$ $(1 \leq i \leq 3)$, $R_2 \in \mathbb{G}_{p_2}$, and $R'_2 \in \mathbb{G}_{p'_2}$ at random, and set $PK = (N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_2, g_3, g = g_1 R_2)$, $SK = (PK, g_1)$, $PK' = (N', \mathbb{G}', \mathbb{G}'_T, \mathbf{e}', g'_2, g'_3, g' = g'_1 R'_2)$, and $SK' = (PK', g'_1)$, where $N = p_1 p_2 p_3$ and $N' = p'_1 p'_2 p'_3$.
4. Pick $s \in \mathbb{Z}_{p_1}$, $s' \in \mathbb{Z}_{p'_1}$, $R_3, R_3 \in \mathbb{G}_{p_3}$, and $R'_3, S'_3 \in \mathbb{G}'_{p'_3}$ at random, and set $T_{x_0} = \langle g_1^s R_3, A_0^s S_3 \rangle$ and $T'_{x_b} = \langle g_1'^{s'} R'_3, X_b^{s'} S'_3 \rangle$. Note that here $\mathcal{B}$ implicitly set $H'(x_0) = X_0$ and $H'(x_1) = X_1$ in the simulation of H' (cf. Step 7).
5. Pick $r \in \mathbb{Z}_N$, $r' \in \mathbb{Z}_{N'}$, $Y_2, Z_2 \in \mathbb{G}_{p_2}$, and $Y'_2, Z'_2 \in \mathbb{G}_{p'_2}$ at random, and set $C_{x_0} = \langle g^r Y_2, A_0^r Z_2 \rangle$ and $C'_{x_b} = \langle g'^{r'} Y'_2, X_b^{r'} Z'_2 \rangle$.
6. Run $\mathcal{A}$ on input $(1^k, PK, PK', (T_{x_0}, C_{x_0}), (T'_{x_b}, C'_{x_b}))$.
7. **RO queries.** To respond to $\mathcal{A}$'s RO queries, $\mathcal{B}$ maintains lists L_H and $L_{H'}$, which are initially empty. Suppose that $\mathcal{A}$ makes an H-query x. If $(x, H(x)) \in L_H$ for some $H(x)$, return $H(x)$ to $\mathcal{A}$. Otherwise, pick a random $H(x) \in \mathbb{G}_{p_1}$, add $(x, H(x))$ to L_H, and return $H(x)$ to $\mathcal{A}$. Next, suppose that $\mathcal{A}$ makes an H'-query x.
   (a) If $x \in \{x_0, x_1\}$, then output a failure message and halt; otherwise, do the following step.
   (b) If $(x, H'(x)) \in L_{H'}$ for some $H'(x)$, return $H'(x)$ to $\mathcal{A}$. Otherwise, pick a random $H'(x) \in \mathbb{G}'_{p'_1}$, add $(x, H'(x))$ to $L_{H'}$, and return $H'(x)$ to $\mathcal{A}$.

8. **Trapdoor queries.** If $\mathcal{A}$ makes a query x to the trapdoor oracle corresponding to SK, return the result of running $\mathsf{Td}^{H}(SK, x)$. Here, the queries to H are responded by the procedure in Step 7. Next, suppose that $\mathcal{A}$ makes query x to the trapdoor oracle corresponding to SK'. If $x \in \{x_0, x_1\}$, then output a failure message and halt; otherwise, return the result of running $\mathsf{Td}^{H'}(SK', x)$.
9. **Guess.** If $\mathcal{A}$ finally outputs $b' \in \{0, 1\}$, then output it as its guess for b.

This completes the description of $\mathcal{B}$. For simplicity, let $\mathsf{Exp}_0 = \mathsf{Exp}^{\text{ind-pkp}}_{\mathcal{A},\Pi_3,\mathcal{X}}(k)$. To analyze the advantage of $\mathcal{B}$, we define the following events in Exp_0.

- $\mathsf{E}_{H'}$ is an event in which $\mathcal{A}$ makes an H'-query x such that $x \in \{x_0, x_1\}$.
- E_t is an event in which $\mathcal{A}$ makes a query $x \in \{x_0, x_1\}$ to the trapdoor oracle $\mathsf{Td}^{H'}(SK', \cdot)$.
- $\mathsf{E} = \mathsf{E}_{H'} \vee \mathsf{E}_t$.

Let Exp_1 be an experiment that is identical with Exp_0 except that if E occurs, it outputs a failure message. Then, from a discussion similar to that in the proof of Proposition 4 (see Appendix D), we can obtain the claimed result.

F Proof of Proposition 6

For given IND-PEKS-CKA adversary $\mathcal{A}$ against Π_3, we construct an algorithm $\mathcal{B}$ that has the underlying feature. $\mathcal{B}$ takes $(1^k, N, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, g_2, g_3, A = g_1^{\alpha}, B = g_1^{\beta}, C = g_1^{\alpha\beta}, g = g_1 R_2, X_0, X_1, D = (g^r Y_2, X_b^r Z_2))$ as input, and its goal is to guess a challenge bit $b \in \{0, 1\}$.

1. Select $i, j \in \{1, 2, \ldots, q_H(k)+2\}$ $(i \neq j)$ randomly and uniformly. Set $PK = (I, g_2, g_3, g)$ and run $\mathcal{A}$ on input $(1^k, PK)$. Note that if $\{i, j\} = \{q_H(k) + 1, q_H(k) + 2\}$ then this implies the case in which $\mathcal{A}$ makes neither H-query x_0 nor x_1 for the challenge query (x_0, x_1, σ) (see Step 2).
2. **RO queries.** Suppose that $\mathcal{A}$ makes an H-query x.
   (a) If this is the i-th H-query, then set $H(x) = X_0$, add $(x, *, H(x))$ to the list L_H, which is initially empty, and return $H(x)$ to $\mathcal{A}$.
   (b) If this is the j-th H-query, then set $H(x) = X_1$, add $(x, *, H(x))$ to the list L_H, and return $H(x)$ to $\mathcal{A}$.
   (c) Otherwise, pick a random $r \in \mathbb{Z}_N$, add $(x, r, H(x) = A^r)$ to the list L_H, and return $H(x)$ to $\mathcal{A}$.
3. **Trapdoor queries.** Suppose that $\mathcal{A}$ makes a trapdoor query x.
   (a) Seek a tuple $(x, r, H(x)) \in L_H$. If there is no such tuple in L_H, then do the same procedure as in Step 2(c) to obtain it.
   (b) Pick $s \in \mathbb{Z}_N$ and $R_3, S_3 \in \mathbb{G}_{p_3}$, and return $T_x = (B^s R_3, C^{rs} S_3)$ to $\mathcal{A}$. Note that since $B^s R_3 = g_1^{s\beta} R_3$ and $C^{rs} S_3 = g_1^{rs\alpha\beta} S_3 = H(x)^{s\beta} S_3$, T_x has the same distribution as the valid trapdoor of x.
4. **Challenge.** If $\mathcal{A}$ makes a challenge query (x_0, x_1, σ), respond with D to $\mathcal{A}$ as an answer to the query.

5. **Guess.** Suppose that $\mathcal{A}$ finally outputs $b' \in \{0,1\}$. If there is no $i \in \{0,1\}$ such that $(x_i, r, H(x_i)) \in L_H$ for some $r \neq *$ then output b' as its guess for b and halt. Otherwise, output a random $b' \in \{0,1\}$ and halt.

This completes the description of $\mathcal{B}$. For the running time, we have $T_{\mathcal{B}}(k) = T_{\mathcal{A}}(k) + O(k^c)$ for some constant $c > 0$. To analyze the advantage of $\mathcal{B}$, we define the following events in the above experiment.

- $\mathsf{Succ}\mathcal{B}$ is an event in which $\mathcal{B}$ succeeds in guessing b.
- E is an event in which there is an $i \in \{0,1\}$ such that $(x_i, r, H(x_i)) \in L_H$ for some $r \neq *$ at Guess stage.

For simplicity, let $\mathsf{Exp} = \mathsf{Exp}^{\text{ind:peks}}_{\mathcal{A},\Pi_3}(k)$. From the specification of $\mathcal{B}$, it completely simulates Exp provided that E does not occur. Hence, we have $\Pr[\mathsf{Succ}\mathcal{B} \mid \neg\mathsf{E}] = \Pr[\mathsf{Exp} = 1]$. On the other hand, $\Pr[\mathsf{Succ}\mathcal{B} \mid \mathsf{E}] = 1/2$. Therefore, we have

$$\begin{aligned}
\mathsf{Adv}^{\text{A-2}}_{\mathcal{B},\mathcal{G}_3}(k) &= \left|\Pr[\mathsf{Succ}\mathcal{B}] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathsf{Succ}\mathcal{B} \mid \neg\mathsf{E}] \cdot \Pr[\neg\mathsf{E}] + \Pr[\mathsf{Succ}\mathcal{B} \mid \mathsf{E}] \cdot \Pr[\mathsf{E}] - \frac{1}{2}\right| \\
&= \Pr[\neg\mathsf{E}] \cdot \left|\Pr[\mathsf{Exp} = 1] - \frac{1}{2}\right| = \Pr[\neg\mathsf{E}] \cdot \mathsf{Adv}^{\text{ind:peks}}_{\mathcal{A},\Pi_3}(k). \qquad (15)
\end{aligned}$$

Since i, j are selected from $\{1, 2, \ldots, q_H(k)\}$ randomly and uniformly, we have

$$\Pr[\neg\mathsf{E}] = \frac{1}{(q_H(k)+1)(q_H(k)+2)}. \qquad (16)$$

From (15) and (16), we can obtain the claimed result.