Generic Construction of Public-key Authenticated Encryption with Keyword Search Revisited: Stronger Security and Efficient Construction[*]

Keita Emura[§]

[§]National Institute of Information and Communications Technology (NICT), Japan.

February 24, 2022

Abstract

Public-key encryption with keyword search (PEKS) does not provide trapdoor privacy, i.e., keyword information is leaked through trapdoors. To prevent this information leakage, public key authenticated encryption with keyword search (PAEKS) has been proposed, where a sender's secret key is required for encryption, and a trapdoor is associated with not only a keyword but also the sender. Liu et al. (ASIACCS 2022) proposed a generic construction of PAEKS based on word-independent smooth projective hash functions (SPHFs) and PEKS. In this paper, we propose a new generic construction of PAEKS. The basic construction methodology is the same as that of the Liu et al. construction, where each keyword is converted into an extended keyword using SPHFs, and PEKS is used for extended keywords. Nevertheless, our construction is more efficient than Liu et al.'s in the sense that we only use one SPHF, but Liu et al. used two SPHFs. In addition, for consistency we considered a security model that is stronger than Liu et al.'s. Briefly, Liu et al. considered only keywords even though a trapdoor is associated with not only a keyword but also a sender. Thus, a trapdoor associated with a sender should not work against ciphertexts generated by the secret key of another sender, even if the same keyword is associated. Our consistency definition considers a multi-sender setting and captures this case. In addition, for indistinguishability against chosen keyword attack (IND-CKA) and indistinguishability against inside keyword guessing attack (IND-IKGA), we use a stronger security model defined by Qin et al. (ProvSec 2021), where an adversary is allowed to query challenge keywords to the encryption and trapdoor oracles. We also highlight several issues associated with the Liu et al. construction in terms of hash functions, e.g., their construction does not satisfy the consistency that they claimed to hold.

1 Introduction

For providing a search functionality against encrypted keyword, public key encryption with keyword search (PEKS) has been proposed by Boneh et al. [7]. As a feasibility result, PEKS can be generically constructed from anonymous identity-based encryption (IBE) [1]. PEKS is briefly explained as follows. A sender encrypts a keyword using a receiver public key. A receiver generates a token to search for a keyword, called trapdoor, using the receiver's secret key. Then, based on the test algorithm, anyone can determine whether a ciphertext is an encryption of a keyword using

[*]An extended abstract appeared at ACM APKC 2022.

a trapdoor. The test algorithm outputs 1 if the two keywords used for encryption and trapdoor generation are the same. In addition to this correctness, consistency is required, with the test algorithm outputting 0 if the two keywords used for encryption and trapdoor generation differ. Moreover, it is required that no information of keyword is leaked from ciphertexts. Unfortunately, PEKS does not provide trapdoor privacy, that is, information of keyword is leaked from trapdoors. More concretely, one can freely generate a ciphertext of an arbitrary-chosen keyword using the receiver's public key. Thus, when one obtains a trapdoor, one can check whether the trapdoor is associated to the keyword via the test algorithm.[1] One way to prevent the keyword guessing attack is to restrict searching, and another way is to restrict encryption.

The former approach is called designated-tester PEKS [3,16–19,31–33].[2] A server, who runs the test algorithm, also has a public key and a secret key pair. A sender, who generates a ciphertext, encrypts a keyword using both a receiver public key and the server public key. Then, the server secret key is required for running the test algorithm, in addition to a trapdoor. The latter approach is called public key authenticated encryption with keyword search (PAEKS) [11, 12, 20, 23, 24, 26, 28–30], and we focus on PAEKS in this paper. A sender, who generates a ciphertext, has a public key and a secret key pair. The sender encrypts a keyword using both a receiver's public key and the sender's secret key. A trapdoor is associated with both a keyword and a sender's public key. That is, the trapdoor only works against ciphertexts generated by the corresponding sender's secret key. Currently, two generic constructions of PAEKS have been proposed [23, 24]. Liu et al. [24] claimed that the construction proposed in [23] does not follow the syntax of PAEKS because it requires a trusted authority to assist users in generating their private keys. More precisely, the setup algorithm outputs a master secret key, and other sender/receiver key generations require the master secret key. Thus, we mainly consider the Liu et al. generic construction [24] in this paper. They employed word-independent smooth projective hash functions (SPHFs) [5,21]. Each keyword is converted into an extended keyword using SPHF, and they employed PEKS for these extended keywords. We revisit their construction methodology in Section 4.

Our Contribution. In this paper, we propose a new generic construction of PAEKS from public key encryption (PKE), word-independent SPHF, pseudorandom function (PRF), and PEKS. The basic construction methodology is the same as that of the Liu et al. construction. Nevertheless, our construction is more efficient than the Liu et al. construction in the sense that we just employ one SPHF but Liu et al. employed two SPHFs. Moreover, for consistency we consider a stronger security model than that of Liu et al. Briefly, they just considered keywords while a trapdoor is associated with not only a keyword but also a sender. So, a trapdoor associated to a sender should not work against ciphertexts generated by a secret key of other sender, even if the same keyword is associated. Our definition considers a multi-sender setting, and captures this case. In addition, for indistinguishability against chosen keyword attack (IND-CKA) and indistinguishability against inside keyword guessing attack (IND-IKGA) (defined in Section 3.2), we employ a stronger security model defined by Qin et al. [30] where an adversary is allowed to query challenge keywords to the encryption and trapdoor oracles (though we modify it in accordance with our syntax).

[1]Boneh, Raghunathan, and Segev proposed function-private IBE [9]. They showed that function-private IBE (with anonymity) can be used for constructing PEKS schemes that are provably keyword private. Concretely, a trapdoor enables to identify encryptions of an underlying keyword, while not revealing any additional information about the keyword *beyond the minimum necessary*, as long as the keyword is sufficiently unpredictable. Owing to the search functionality of PEKS, it is inevitable to leak information of keyword as mentioned.

[2]In PEKS, a receiver needs to send a trapdoor to the server via a secure channel since anyone who has a trapdoor can run the test algorithm. On the other hand, the server secret key is required to run the test algorithm in designated-tester PEKS, and thus no secure channel between the server and the receiver is required. Therefore, designated-tester PEKS is alternatively called secure channel free PEKS (SCF-PEKS).

We also point out there are several issues in the Liu et al. proposal regarding their construction and security models (See Section 4.2 in detail). The main issue is related to hash functions they employed. First, their construction does not satisfy the consistency requirement that they claimed to hold. Second, they assume that a hash value does not leak input information, but it is not guaranteed by the one-wayness of hash functions. These issues may be solved by introducing other consistency definitions and by assuming that the underlying hash functions are modeled as random oracles. Besides these issues, they considered a weak security model, as previously mentioned.

We introduce a designated-receiver setting, where the sender key generation algorithm takes as input a receiver's public key. The setting allows us to remove a trusted setup assumption (In other words, if we assume a trusted setup, we do not have to introduce the designated-receiver setting). See Section 3.1 in detail.

Disclaimer. We checked the ePrint version of Liu et al. paper (Version 3, posted on 23-Nov-2021) [24] that is the same as their AsiaCCS 2022 version as they declared.

2 Preliminaries

Notations. For a positive integer $n \in \mathbb{N}$, we write $[1, n] = \{1, 2, \ldots, n\}$. If A is a probabilistic algorithm, $y \leftarrow A(x; r)$ denotes the operation of running A on an input x and a randomness r, and letting y be the output. We omit r when it is not necessary to specify. $x \xleftarrow{\$} S$ denotes choosing an element x from a finite set S uniformly at random. For a security parameter λ, $\mathsf{negl}(\lambda)$ is a negligible function where for any $c > 0$, there exists an integer I such that $\mathsf{negl}(\lambda) < 1/\lambda^c$ for all $\lambda > I$.

2.1 Pseudorandom Functions (PRFs)

Let m and ℓ be polynomial and λ be a security parameter. A pseudorandom function (PRF) is a family of functions $\mathsf{PRF} = \{F_K : \{0,1\}^{m(\lambda)} \to \{0,1\}^{\ell(\lambda)}\}$ where $K \in \{0,1\}^\lambda$.

Definition 1 (Pseudo-randomness). *We say that* PRF *is pseudo-random if for all probabilistic polynomial-time (PPT) adversaries* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{pseudo\text{-}random}}_{\mathsf{PRF},\mathcal{A}}(\lambda) := |\Pr[K \xleftarrow{\$} \{0,1\}^\lambda : \mathcal{A}^{F_K(\cdot)}(1^\lambda) = 1] - \Pr[R \xleftarrow{\$} \mathcal{RF} : \mathcal{A}^{R(\cdot)}(1^\lambda) = 1]|$ *is negligible in the security parameter* λ *where* $\mathcal{RF}$ *is a set of random functions mapping* $m(\lambda)$ *bits to* $\ell(\lambda)$ *bits.*

2.2 Public Key Encryption (PKE)

A PKE scheme PKE consists of three algorithms $(\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$. The key generation algorithm $\mathsf{PKE.KeyGen}$ takes a security parameter λ as input, and outputs a public key $\mathsf{pk}_{\mathsf{PKE}}$ and a secret key $\mathsf{dk}_{\mathsf{PKE}}$. The encryption algorithm takes $\mathsf{pk}_{\mathsf{PKE}}$ and a plaintext $M \in \mathcal{MS}$ as input, where $\mathcal{MS}$ is a message space, and outputs a ciphertext $\mathsf{ct}_{\mathsf{PKE}}$. When we need to explicitly treat the randomness ρ for encryption, we denote $\mathsf{ct}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, M; \rho)$. The decryption algorithm $\mathsf{PKE.Dec}$ takes $\mathsf{dk}_{\mathsf{PKE}}$ and $\mathsf{ct}_{\mathsf{PKE}}$ as input, and outputs M. For correctness, it is required that for any security parameter λ, any $(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and any plaintext $M \in \mathcal{MS}$, $\Pr[\mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{PKE}}, \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, M)) = M] = 1 - \mathsf{negl}(\lambda)$ holds. We also require that the standard indistinguishability against chosen plaintext attack (IND-CPA) holds. Let state be state information that $\mathcal{A}$ can preserve any information, and state is used for transferring state information to the other stage.

Definition 2 (IND-CPA)**.** *For all PPT adversaries $\mathcal{A}$, we define the following experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda):\\
&\quad (\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)\\
&\quad (M_0^*, M_1^*, \mathsf{state}) \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{PKE}})\\
&\qquad s.t.\ M_0^*, M_1^* \in \mathcal{MS} \wedge M_0^* \neq M_1^* \wedge |M_0^*| = |M_1^*|\\
&\quad b \xleftarrow{\$} \{0,1\};\ \mathsf{ct}^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, M_b^*)\\
&\quad b' \leftarrow \mathcal{A}(\mathsf{ct}^*_{\mathsf{PKE}}, \mathsf{state})\\
&\quad \textit{If } b = b' \textit{ then output } 1 \textit{ and } 0 \textit{ otherwise.}
\end{aligned}
$$

We say that a PKE scheme PKE *is IND-CPA secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{A}}(\lambda) = 1] = \mathsf{negl}(\lambda)$.

2.3 Word-independent Smooth Projective Hash Functions (SPHFs)

Cramer and Shoup proposed hash proof systems (HPSs) [13], which are special kind of non-interactive zero-knowledge proof systems for a language, for constructing PKE. Later, several applications of HPSs have been considered such as password-based authenticated key exchange (PAKE) (e.g., [5, 21]). In this paper, we employ word-independent smooth projective hash functions (SPHFs) (which are also called KV-SPHF [5] in reference to [21]). We introduce the definition given by Benhamouda et al. [6].

Definition 3 (Languages)**.** *Let* $\mathsf{Setup.lpar}$ *be a PPT algorithm that takes a security parameter* λ *as input, and outputs* $(\mathsf{lpar}, \mathsf{ltrap})$ *where* lpar *is a parameter and* ltrap *is a trapdoor.* $\mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}$ *is a language indexed by* $(\mathsf{lpar}, \mathsf{ltrap})$ *together with an NP language indexed by* lpar $\widetilde{\mathscr{L}}_{\mathsf{lpar}}$*, with witness relation* $\widetilde{\mathscr{R}}_{\mathsf{lpar}}$ *such that* $\widetilde{\mathscr{L}}_{\mathsf{lpar}} = \{\chi \in \mathcal{X}_{\mathsf{lpar}} \mid \exists w \ s.t.\ \widetilde{\mathscr{R}}_{\mathsf{lpar}}(\chi, w) = 1\} \subseteq \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}} \subseteq \mathcal{X}_{\mathsf{lpar}}$*. We denote* $(\widetilde{\mathscr{L}}_{\mathsf{lpar}}, \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}, \mathcal{X}_{\mathsf{lpar}})_{\mathsf{lpar},\mathsf{ltrap}}$ *as a family of languages.*

Next, we define languages of ciphertexts. Benhamouda et al. [6] introduced languages of ciphertexts for a labeled PKE scheme. More precisely, Benhamouda et al. defined the languages for a IND-CCA2 secure labeled PKE scheme that is converted from (a simplified variant of) the tag-IND-CCA2 secure Micciancio-Peikert PKE scheme [25] using the Dolev-Dwork-Naor (DDN) transformation [14]. Then, they also showed that the tag PKE scheme is IND-CPA secure when the tag of the ciphertext is known in advance or is constant, and proposed a word-independent SPHF for IND-CPA ciphertexts. Since we need to employ the word-independent SPHF, we focus on their languages of IND-CPA ciphertexts. Since the tag/label can be a constant value, it can be included in $\mathsf{pk}_{\mathsf{PKE}}$ in advance, and then we do not have to consider tag/label anymore. Thus, we define languages of ciphertexts for a standard PKE scheme (defined in Section 2.2). Moreover, Benhamouda et al. focused on the languages of ciphertexts of 0, we also consider such ciphertexts as follows.

Definition 4 (Languages of Ciphertexts [6])**.** *Let* $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ *be a PKE scheme. The* $\mathsf{Setup.lpar}$ *algorithm is set as* $\mathsf{PKE.KeyGen}$*, and* $(\mathsf{lpar}, \mathsf{ltrap}) = (\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}})$*. Then, languages of ciphertexts are defined as* $\widetilde{\mathscr{L}}_{\mathsf{pk}_{\mathsf{PKE}}} = \{\mathsf{ct}_{\mathsf{PKE}} \mid \exists \rho \ s.t.\ \mathsf{ct}_{\mathsf{PKE}} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, 0; \rho)\} \subseteq \mathscr{L}_{\mathsf{pk}_{\mathsf{PKE}},\mathsf{dk}_{\mathsf{PKE}}} = \{\mathsf{ct}_{\mathsf{PKE}} \mid \mathsf{PKE.Dec}(\mathsf{dk}_{\mathsf{PKE}}, \mathsf{ct}_{\mathsf{PKE}}) = 0\}$ *where* $\mathcal{X}_{\mathsf{pk}_{\mathsf{PKE}}}$ *is a set of valid ciphertexts generated by* $\mathsf{pk}_{\mathsf{PKE}}$*. Here,* $\widetilde{\mathscr{R}}_{\mathsf{lpar}}(\mathsf{ct}_{\mathsf{PKE}}, \rho) = 1$ *iff* $\mathsf{ct}_{\mathsf{PKE}} = \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, 0; \rho)$.

In the actual Benhamouda et al. construction, dual-Regev ciphertexts of 0 is considered: $\mathbf{c} = \mathbf{A}\mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m$ where $\mathbf{A} \in \mathbb{Z}_q^{m \times n}$ is a public matrix, $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e} \in \mathbb{Z}_q^m$ are the randomness. If $\mathbf{e}$ is supposed to be small, then $\mathbf{c}$ is close to the q-ary lattice Λ generated by $\mathbf{A}$. On the other hand, if $\mathbf{c}$ is an encryption of 1, then $\mathbf{c}$ is far from the q-ary lattice Λ generated by $\mathbf{A}$. Thus, whether a word $\mathbf{c}$ belongs to $\widetilde{\mathscr{L}}_{\mathsf{pk}_{\mathsf{PKE}}}$ (i.e., a ciphertext of 0) or $\mathscr{X}_{\mathsf{pk}_{\mathsf{PKE}}} \setminus \mathscr{L}_{\mathsf{pk}_{\mathsf{PKE}}}$ (i.e., a ciphertext whose decryption result is 1) is indistinguishable owing to the IND-CPA security of PKE. Obviously, the membership in $\mathscr{L}_{\mathsf{pk}_{\mathsf{PKE}},\mathsf{dk}_{\mathsf{PKE}}}$ can be checked in polynomial time by $\mathsf{dk}_{\mathsf{PKE}}$.

Next, we define the syntax of word-independent SPHF. Here, the term "word independent" means that the ProjKG algorithm does not take a word $\chi \in \mathscr{X}_{\mathsf{lpar}}$ as an input, and the smoothness (defined below) holds even if the word is chosen adaptively after seeing the projection key (thus, the smoothness is called adaptive smoothness).

Definition 5 (Syntax of Word-independent SPHF [6]). *Let $(\widetilde{\mathscr{L}}_{\mathsf{lpar}}, \mathscr{L}_{\mathsf{lpar},\mathsf{ltrap}}, \mathscr{X}_{\mathsf{lpar}})_{\mathsf{lpar},\mathsf{ltrap}}$ be a family of languages. A word-independent SPHF* $\mathsf{WI\text{-}SPHF}$ *for these languages consists of four algorithms* $(\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ *defined as follows. Let λ be a security parameter used for running the* $\mathsf{Setup.lpar}$ *algorithm.*

HashKG: *The hash key generation algorithm takes* lpar *as input, and outputs a hashing key* hk.

ProjKG: *The projection key derivation algorithm takes* hk *and* lpar *as input, and outputs a projection key* hp.

Hash: *The hash algorithm takes* hk, lpar, *and a word $\chi \in \mathscr{X}_{\mathsf{lpar}}$ as input, and outputs a hash value* $\mathsf{H} \in \{0,1\}^\nu$ *for some positive integer $\nu = \Omega(\lambda)$.*

$\mathsf{ProjHash}$: *The projected hash algorithm takes* hp, lpar, *χ, and the witness w for the word $\chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}}$ as input, and outputs a projected hash value* $\mathsf{pH} \in \{0,1\}^\nu$.

Definition 6 (Correctness [6]). *For any security parameter λ, let* $(\mathsf{lpar}, \mathsf{ltrap}) \leftarrow \mathsf{Setup.lpar}(1^\lambda)$. *With overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$, *for any $\chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}}$ and associated witness w,* $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi)$ *is approximately determined by* $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar})$ *relative to the Hamming distance* HD *where* $\Pr[\mathsf{HD}(\mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, \chi), \mathsf{ProjHash}(\mathsf{hp}, \mathsf{lpar}, \chi, w) > \epsilon \cdot \nu] = \mathsf{negl}(\lambda)$. *Here, the probability is taken over the choice of* $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar})$ *and the randomness of* Hash *and* $\mathsf{ProjHash}$.[3] *Then, we say that* $\mathsf{WI\text{-}SPHF}$ *is approximately ϵ-correct. Moreover, we say that* $\mathsf{WI\text{-}SPHF}$ *is statistically correct if it is 0-correct.*

We remark that Benhamouda et al. first constructed a word-independent bit-PHF for languages of IND-CPA ciphertexts, where the hash value is just a bit (i.e., $\nu = 1$). Next they showed that a word-independent SPHF, with the output length $\{0,1\}^\nu$ where $\nu = \Omega(\lambda)$, can be constructed from a word-independent bit-PHF generically (Lemma B.4 [6]). Here, the original word-independent bit-PHF is supposed to be statistically correct (if it is approximate ϵ-correct, then the converted SPHF is not word-independent due to additional error correcting codes, even the underlying bit-PHF is word-independent). Thus, we employ correctness rather than approximate correctness. Benhamouda et al. showed how to construct a bit-PHF with statistical correctness (Lemma 4.1 [6]).[4]

Definition 7 (Adaptive Smoothness [6]). *For any security parameter λ, let* $(\mathsf{lpar}, \mathsf{ltrap}) \leftarrow \mathsf{Setup.lpar}(1^\lambda)$. *With overwhelming probability over the randomness of* $\mathsf{Setup.lpar}$, *for all function f onto $\mathscr{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$,*

[3] As mentioned by Benhamouda et al., they considered probabilistic Hash and $\mathsf{ProjHash}$ algorithms.

[4] Li and Wang [22] proposed a word-independent approximate SPHF without employing error correcting codes. While this may be employed in our construction, however, we prefer correctness because it is more desirable.

the following two distributions have statistical distance negligible in λ:

$$\{(\mathsf{lpar}, f(\mathsf{hp}), \mathsf{hp}, \mathsf{H}) | \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar}), \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}), \mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f(\mathsf{hp}))\}$$

and

$$\{(\mathsf{lpar}, f(\mathsf{hp}), \mathsf{hp}, \mathsf{H}) | \mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{lpar}), \mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{lpar}), \mathsf{H} \overset{\$}{\leftarrow} \{0,1\}^{\nu}\}$$

2.4 Public-key Encryption with Keyword Search (PEKS)

In this section, we introduce the definitions of PEKS given in [1].

Definition 8 (Syntax of PEKS [1])**.** *A PEKS scheme* PEKS *consists of four algorithms* $(\mathsf{PEKS.KG}, \mathsf{PEKS.Enc}, \mathsf{PEKS.Trapdoor}, \mathsf{PEKS.Test})$ *defined as follows.*

$\mathsf{PEKS.KG}$**:** *The key generation algorithm takes a security parameter λ as input, and outputs a public key* $\mathsf{pk}_{\mathsf{PEKS}}$ *and a secret key* $\mathsf{sk}_{\mathsf{PEKS}}$*. We assume that* $\mathsf{pk}_{\mathsf{PEKS}}$ *implicitly contains the keyword space* $\mathcal{KS}$*.*

$\mathsf{PEKS.Enc}$**:** *The keyword encryption algorithm takes* $\mathsf{pk}_{\mathsf{PEKS}}$ *and a keyword $kw \in \mathcal{KS}$ as input, and outputs a ciphertext* $\mathsf{ct}_{\mathsf{PEKS}}$*.*

$\mathsf{PEKS.Trapdoor}$**:** *The trapdoor algorithm takes* $\mathsf{pk}_{\mathsf{PEKS}}$*,* $\mathsf{sk}_{\mathsf{PEKS}}$*, and a keyword $kw \in \mathcal{KS}$ as input, and outputs a trapdoor* td_{kw}*.*

$\mathsf{PEKS.Test}$**:** *The test algorithm takes* $\mathsf{ct}_{\mathsf{PEKS}}$ *and* td_{kw} *as input, and outputs 1 or 0.*

In accordance with the definition given in [1], correctness and consistency are separately defined. Briefly, for a ciphertext of a keyword kw and a trapdoor of a keyword kw', the former guarantees that the $\mathsf{PEKS.Test}$ algorithm outputs 1 if $kw = kw'$, and the latter guarantees that the $\mathsf{PEKS.Test}$ algorithm outputs 0 if $kw \neq kw'$. We emphasize that we employ computational consistency, i.e., consistency holds against computationally bounded adversaries.

Definition 9 (Correctness [1])**.** *For any security parameter λ, any key pairs* $(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$ *and any keyword $kw \in \mathcal{KS}$, let* $\mathsf{ct}_{\mathsf{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk}_{\mathsf{PEKS}}, kw)$ *and* $\mathsf{td}_{kw} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, kw)$*. Then* $\Pr[\mathsf{PEKS.Test}(\mathsf{ct}_{\mathsf{PEKS}}, \mathsf{td}_{kw}) = 1] = 1 - \mathsf{negl}(\lambda)$ *holds.*

Definition 10 (Computational Consistency [1])**.** *For all PPT adversaries $\mathcal{A}$, we define the following experiment:*

```
$\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{PEKS},\mathcal{A}}(\lambda)$ :
    $(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$
    $(kw, kw') \leftarrow \mathcal{A}(\mathsf{pk}_{\mathsf{PEKS}})$ s.t. $kw, kw' \in \mathcal{KS} \wedge kw \neq kw'$
    $\mathsf{ct}_{\mathsf{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk}_{\mathsf{PEKS}}, kw)$
    $\mathsf{td}_{kw'} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, kw')$
    If $\mathsf{PEKS.Test}(\mathsf{ct}_{\mathsf{PEKS}}, \mathsf{td}_{kw'}) = 1$
      then output $1$ and $0$ otherwise.
```

We say that a PEKS scheme PEKS *is consistent if the advantage* $\mathsf{Adv}^{\mathsf{consist}}_{\mathsf{PEKS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{PEKS},\mathcal{A}}(\lambda) = 1]$ *is negligible in the security parameter λ.*

Next, we define ciphertext indistinguishability against chosen keyword attack (IND-CKA)[5] which guarantees that no information of keyword is leaked from ciphertexts.

Definition 11 (IND-CKA [1]). *For all PPT adversaries $\mathcal{A}$, we define the following experiment:*

$$\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEKS},\mathcal{A}}(\lambda, n):\\
&\quad (\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)\\
&\quad (kw_0^*, kw_1^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_T(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, \cdot)}(\mathsf{pk}_{\mathsf{PEKS}})\\
&\quad\quad s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge kw_0^* \neq kw_1^*\\
&\quad b \xleftarrow{\$} \{0,1\};\ \mathsf{ct}^*_{\mathsf{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk}_{\mathsf{PEKS}}, kw_b^*)\\
&\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_T(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, \cdot)}(\mathsf{state}, \mathsf{ct}^*_{\mathsf{PEKS}})\\
&\quad \text{If } b = b' \text{ then output } 1 \text{ and } 0 \text{ otherwise.}
\end{aligned}$$

$\mathcal{O}_T$ takes $kw \in \mathcal{KS}$ as input, and returns $\mathsf{td}_{kw} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, kw)$. Here $kw \notin \{kw_0^*, kw_1^*\}$. We say that a PEKS scheme PEKS is IND-CKA secure if the advantage $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEKS},\mathcal{A}}(\lambda, n) := \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEKS},\mathcal{A}}(\lambda, n) = 1]$ is negligible in the security parameter λ.

We emphasize that PEKS does not provide trapdoor privacy, i.e., information of kw is leaked from td_{kw}. Actually, for some trapdoor td_{kw}, anyone can compute $\mathsf{ct}_{\mathsf{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk}_{\mathsf{PEKS}}, kw')$ for any $kw' \in \mathcal{KS}$, and then anyone can check whether $kw = kw'$ or not by running $\mathsf{PEKS.Test}(\mathsf{ct}_{\mathsf{PEKS}}, \mathsf{td}_{kw})$.

3 Definitions of Designated-Receiver Multi-Sender PAEKS

3.1 Designated-Receiver Setting

In the previous definition [11,12,20,24,26,28–30], a setup algorithm is defined that takes a security parameter as input, and outputs a public parameter pp. Then, two key generation algorithms, $\mathsf{PAEKS.KG_R}$ and $\mathsf{PAEKS.KG_S}$, are defined which take as input pp, and output a key pair, respectively. In our definition, the sender key generation algorithm $\mathsf{PAEKS.KG_S}$ takes a receiver public key $\mathsf{pk_R}$ as input, i.e., our definition captures a designated-receiver setting.

One may think that this setting restricts the flexibility of key generations. However, a sender needs to designate a receiver before the sender encrypts a keyword. Then, due to the functionality of PAEKS, the sender needs to use its own secret key $\mathsf{sk_S}$. Since the sender has designated the receiver, the designated-receiver key generation does not restrict encryption. Of course, the designated-receiver setting restricts the order of searching, i.e., no trapdoor can be generated before the corresponding sender's public key is generated. Beyond this restriction, there is a merit of the designated-receiver setting that can avoid introducing a trusted setup. It is required in the Liu et al. construction [24] that no one knows a decryption key $\mathsf{dk}_{\mathsf{PKE}}$ of the underlying PKE scheme because $\mathsf{dk}_{\mathsf{PKE}} = \mathsf{ltrap}$ can be used to break the underlying membership problem. Thus, the setup algorithm outputs $\mathsf{pk}_{\mathsf{PKE}}$ only. Liu et al. introduced a hash function that generates $\mathsf{pk}_{\mathsf{PKE}}$ (it seems to guarantee that there is no corresponding decryption key). However, how to construct such a hash function is unclear (we will discuss it in detail in Section 4.2). In the designated-receiver setting, the $\mathsf{PAEKS.KG_R}$ algorithm simply runs $(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and $\mathsf{dk}_{\mathsf{PKE}}$ can be regarded as a secret key of the receiver. Then, we do not have to consider how to erase $\mathsf{dk}_{\mathsf{PKE}}$ anymore (we will show that giving $\mathsf{dk}_{\mathsf{PKE}}$ to a receiver does not affect security). In other words, if we assume a trusted setup, where a setup algorithm runs $(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, erase

[5] In [1], this security notion is called PEKS-IND-CPA.

$\mathsf{dk}_{\mathsf{PKE}}$ and outputs $\mathsf{pk}_{\mathsf{PKE}}$ only as a public parameter, we do not have to introduce the designated-receiver setting. However, it would be better to clarify who can know $\mathsf{dk}_{\mathsf{PKE}}$. In this sense, our setting is more desirable.

3.2 Definitions of PAEKS

Owing to the functionality of PAEKS, a trapdoor is associated with not only a keyword kw but also a sender by indicating the sender's public key pk_{S}. Thus, we explicitly denote a trapdoor $\mathsf{td}_{\mathsf{S},kw}$. Because we consider multiple senders, the $\mathsf{PAEKS.KG}_{\mathsf{S}}$ algorithm is run by each sender. For simplicity, we assume that there are n senders, and we denote the i-th sender's key pair as $(\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]})$ where $i \in [1,n]$ is the sender index, and denote $\mathsf{td}_{\mathsf{S}[i],kw}$ as a trapdoor generated by indicating $\mathsf{pk}_{\mathsf{S}[i]}$ and kw. We use $(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}})$ and $\mathsf{td}_{\mathsf{S},kw}$ when no sender index is explicitly appeared.

Definition 12 (Syntax of Designated-Receiver Multi-Sender PAEKS). *A PAEKS scheme* PAEKS *consists of five algorithms* $(\mathsf{PAEKS.KG}_{\mathsf{R}}, \mathsf{PAEKS.KG}_{\mathsf{S}}, \mathsf{PAEKS.Enc}, \mathsf{PAEKS.Trapdoor}, \mathsf{PEKS.Test})$ *defined as follows.*

$\mathsf{PAEKS.KG}_{\mathsf{R}}$: *The receiver key generation algorithm takes a security parameter λ as input, and outputs a public key pk_{R} and a secret key sk_{R}. We assume that pk_{R} implicitly contains the keyword space $\mathcal{KS}$.*

$\mathsf{PAEKS.KG}_{\mathsf{S}}$: *The sender key generation algorithm takes pk_{R} as input, and outputs a public key pk_{S} and a secret key sk_{S}.*

$\mathsf{PAEKS.Enc}$: *The keyword encryption algorithm takes pk_{R}, pk_{S}, sk_{S}, and a keyword $kw \in \mathcal{KS}$ as input, and outputs a ciphertext $\mathsf{ct}_{\mathsf{PAEKS}}$.*

$\mathsf{PAEKS.Trapdoor}$: *The trapdoor algorithm takes pk_{R}, pk_{S}, sk_{R}, and a keyword $kw \in \mathcal{KS}$ as input, and outputs a trapdoor $\mathsf{td}_{\mathsf{S},kw}$.*

$\mathsf{PAEKS.Test}$: *The test algorithm takes $\mathsf{ct}_{\mathsf{PAEKS}}$ and $\mathsf{td}_{\mathsf{S},kw}$ as input, and outputs 1 or 0.*

Definition 13 (Correctness). *For any security parameter λ, any key pairs $(\mathsf{pk}_{\mathsf{R}}, \mathsf{sk}_{\mathsf{R}}) \leftarrow \mathsf{PAEKS.KG}_{\mathsf{R}}(1^\lambda)$ and $(\mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}) \leftarrow \mathsf{PAEKS.KG}_{\mathsf{S}}(\mathsf{pk}_{\mathsf{R}})$, and any keyword $kw \in \mathcal{KS}$, let $\mathsf{ct}_{\mathsf{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, kw)$ and $\mathsf{td}_{\mathsf{S},kw} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{R}}, kw)$. Then $\Pr[\mathsf{PAEKS.Test}(\mathsf{ct}_{\mathsf{PAEKS}}, \mathsf{td}_{\mathsf{S},kw}) = 1] = 1 - \mathsf{negl}(\lambda)$ holds.*

Next, we define consistency. Previous works employ the following definition: as in the same setting of the correctness, except for two keywords $kw, kw' \in \mathcal{KS}$,

$$\Pr[\mathsf{PAEKS.Test}(\mathsf{ct}_{\mathsf{PAEKS}}, \mathsf{td}_{\mathsf{S},kw'}) = 0] = 1 - \mathsf{negl}(\lambda)$$

holds where $\mathsf{ct}_{\mathsf{PAEKS}}$ is a ciphertext of kw, $\mathsf{td}_{\mathsf{S},kw'}$ is a trapdoor of kw', and $kw \neq kw'$. Before giving our definition, we point out two problems of this previous definition.

1. It captures statistical consistency where it requires to hold even against computationally unbounded adversaries. However, the previous schemes do not satisfy this definition because they used a collision-resistant hash function for providing consistency (i.e., if $kw \neq kw'$, then its hash values are supposed to be different). That is, there "exists" kw and kw' such that its hash values are the same but $kw \neq kw'$ and computationally unbounded adversaries can find them. Thus, we need to consider computationally bounded adversaries. Of course, there is a room for providing statistical consistency as in a PEKS scheme given by Abdalla et al. [1].

We remark that the construction methodology of the statistically consistent Abdalla et al. PEKS scheme completely contradicts trapdoor privacy because a keyword itself is contained in a trapdoor (this setting does not contradict to provide IND-CKA). Thus it seems difficult to construct a statistically consistent PAEKS scheme with trapdoor privacy (though we do not exclude the possibility). Thus, in this paper we define computational consistency that considers PPT adversaries.

2. The previous definitions only considered keywords, however, a trapdoor is associated with not only a keyword kw but also a sender by indicating the sender's public key $\mathsf{pk_S}$. Thus, for $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_{S[0]}}, \mathsf{sk_{S[0]}}, kw)$ and $\mathsf{td}_{\mathsf{S}[1],kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_{S[1]}}, \mathsf{sk_R}, kw')$, $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}_{\mathsf{S}[1],kw'}) = 0$ should be hold even if $kw = kw'$. This case is not captured in the previous definition. We consider this case by setting $(kw, i) \neq (kw', j)$ in the experiment.

Definition 14 (Computational Consistency)**.** *For all PPT adversaries $\mathcal{A}$, we define the following experiment:*

```
$\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda)$ :
    $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(1^\lambda)$
    $(\mathsf{pk_{S[0]}}, \mathsf{sk_{S[0]}}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})$
    $(\mathsf{pk_{S[1]}}, \mathsf{sk_{S[1]}}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})$
    $(kw, kw', i, j) \leftarrow \mathcal{A}(\mathsf{pk_R}, \mathsf{pk_{S[0]}}, \mathsf{pk_{S[1]}})$
        $s.t.\ kw, kw' \in \mathcal{KS} \wedge i, j \in \{0,1\} \wedge (kw, i) \neq (kw', j)$
    $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}, kw)$
    $\mathsf{td}_{\mathsf{S}[j],kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[j]}, \mathsf{sk_R}, kw')$
    If $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}_{\mathsf{S}[j],kw'}) = 1$
      then output $1$ and $0$ otherwise.
```

We say that a PAEKS scheme PAEKS *is consistent if the advantage* $\mathsf{Adv}^{\mathsf{consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1]$ *is negligible in the security parameter λ.*

Next, we define two indistinguishability notions. First, we define ciphertext privacy by formalizing indistinguishability against chosen keyword attack (IND-CKA) which guarantees that no information of keyword is leaked from ciphertexts (as in PEKS). In the previous IND-CKA definition, two oracles are defined, an encryption oracle $\mathcal{O}_C$ and a trapdoor oracle $\mathcal{O}_T$. Then, two keywords, kw_0^* and kw_1^* are chosen by the adversary $\mathcal{A}$, and the challenge ciphertext $\mathsf{ct}^*_{\mathsf{PAEKS}}$ is computed by either kw_0^* or kw_1^*. Here, we need to consider what values can be input to these oracles. To exclude the trivial case, we need to restrict $\mathcal{A}$ to obtaining a ciphertext that can be used for distinguishing between kw_0^* and kw_1^*. In other words, $\mathcal{A}$ should be allowed to obtain any trapdoor excluding the above case, e.g., $\mathsf{td}_{\mathsf{S},kw_0^*} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw_0^*)$ if $\mathsf{sk_S}$ is not used for generating the challenge ciphertext. In almost previous definition, $\mathcal{A}$ is not allowed to input kw_0^* and kw_1^* to $\mathcal{O}_C$ and $\mathcal{O}_T$. Qin et al. [30] have improved this restriction where kw_0^* and kw_1^* can be inputs. For $\mathcal{O}_T$, the receiver is fixed and it matches the multi-sender setting. For $\mathcal{O}_C$, the sender is fixed, and it does not consider the multi-sender setting. Thus, we modify the definition of $\mathcal{O}_C$ following the multi-sender setting.

Definition 15 (IND-CKA). *For all PPT adversaries $\mathcal{A}$, we define the following experiment:*

$$\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda, n):\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(1^\lambda)\\
&\quad \textit{For } i \in [1,n],\ (\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})\\
&\quad (kw_0^*, kw_1^*, i^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_C(\mathsf{pk_R},\cdot,\cdot),\mathcal{O}_T(\mathsf{pk_R},\cdot,\mathsf{sk_R},\cdot)}(\mathsf{pk_R}, \{\mathsf{pk}_{\mathsf{S}[i]}\}_{i\in[1,n]})\\
&\qquad s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^* \wedge i^* \in [1,n]\\
&\quad b \xleftarrow{\$} \{0,1\};\ \mathsf{ct}^*_{\mathsf{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i^*]}, \mathsf{sk}_{\mathsf{S}[i^*]}, kw_b^*)\\
&\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_C(\mathsf{pk_R},\cdot,\cdot),\mathcal{O}_T(\mathsf{pk_R},\cdot,\mathsf{sk_R},\cdot)}(\mathsf{state}, \mathsf{ct}^*_{\mathsf{PAEKS}})\\
&\quad \textit{If } b = b' \textit{ then output } 1 \textit{ and } 0 \textit{ otherwise.}
\end{aligned}$$

$\mathcal{O}_C$ takes $kw \in \mathcal{KS}$ and $i \in [1,n]$ as input, and returns $\mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}, kw)$*. Here, there is no restriction. $\mathcal{O}_T$ takes $kw \in \mathcal{KS}$ and $i \in [1,n]$ as input, and returns* $\mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk_R}, kw)$*. Here $(kw, i) \notin \{(kw_0^*, i^*), (kw_1^*, i^*)\}$. We say that a PAEKS scheme* PAEKS *is IND-CKA secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda, n) := \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda, n) = 1]$ *is negligible in the security parameter λ.*

Next, we define trapdoor privacy by formalizing indistinguishability against inside keyword guessing attack (IND-IKGA) which guarantees that no information of keyword is leaked from trapdoors. To exclude the trivial case, we need to restrict that $\mathcal{A}$ obtains a ciphertext that can be used for distinguishing kw_0^* or kw_1^*. Again we revisit Qin et al.'s definition [30]. For $\mathcal{O}_T$, the receiver is fixed, and it matches the multi-sender setting. For $\mathcal{O}_C$ the sender is fixed and it does not capture the multi-sender setting. So, we modify the definition of $\mathcal{O}_C$ in accordance with the multi-sender setting.

Definition 16 (IND-IKGA). *For all PPT adversaries $\mathcal{A}$, we define the following experiment:*

$$\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}IKGA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda, n):\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(1^\lambda)\\
&\quad \textit{For } i \in [1,n],\ (\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})\\
&\quad (kw_0^*, kw_1^*, i^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}_C(\mathsf{pk_R},\cdot,\cdot),\mathcal{O}_T(\mathsf{pk_R},\cdot,\mathsf{sk_R},\cdot)}(\mathsf{pk_R}, \{\mathsf{pk}_{\mathsf{S}[i]}\}_{i\in[1,n]})\\
&\qquad s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^* \wedge i^* \in [1,n]\\
&\quad b \xleftarrow{\$} \{0,1\};\ \mathsf{td}^*_{\mathsf{S}[i^*],kw_b^*} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i^*]}, \mathsf{sk_R}, kw_b^*)\\
&\quad b' \leftarrow \mathcal{A}^{\mathcal{O}_C(\mathsf{pk_R},\cdot,\cdot),\mathcal{O}_T(\mathsf{pk_R},\cdot,\mathsf{sk_R},\cdot)}(\mathsf{state}, \mathsf{td}^*_{\mathsf{S}[i^*],kw_b^*})\\
&\quad \textit{If } b = b' \textit{ then output } 1 \textit{ and } 0 \textit{ otherwise.}
\end{aligned}$$

$\mathcal{O}_C$ takes $kw \in \mathcal{KS}$ and $i \in [1,n]$ as input, and returns $\mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}, kw)$*. Here, $(kw, i) \notin \{(kw_0^*, i^*), (kw_1^*, i^*)\}$. $\mathcal{O}_T$ takes $kw \in \mathcal{KS}$ and $i \in [1,n]$ as input, and returns* $\mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk_R}, kw)$*. Here $(kw, i) \notin \{(kw_0^*, i^*), (kw_1^*, i^*)\}$. We say that a PAEKS scheme* PAEKS *is IND-IKGA secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}IKGA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda, n) := \Pr[\mathsf{Exp}^{\mathsf{IND\text{-}IKGA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda, n) = 1]$ *is negligible in the security parameter λ.*

Relation between Our Definitions and Multi-Ciphertext/Trapdoor Indistinguishability. Qin et al. [29] considered multi-ciphertext indistinguishability (MCI) where in the IND-CKA experiment $\mathcal{A}$ declares two keyword vectors $(kw^*_{0,1}, \ldots, kw^*_{0,N})$ and $(kw^*_{1,1}, \ldots, kw^*_{1,N})$ for some N, and the challenger returns the challenge ciphertexts of $kw^*_{b,i}$ for $i \in [1, N]$. As mentioned in [30], if the encryption oracle $\mathcal{O}_C$ has no restriction (i.e., any input is allowed), then IND-CKA implies MCI. Thus, our IND-CKA definition provides MCI security.

Similarly, Pan and Li [28] considered multi-trapdoor indistinguishability (MTI) where in the IND-IKGA experiment $\mathcal{A}$ declares two keyword vectors $(kw^*_{0,1}, \ldots, kw^*_{0,N})$ and $(kw^*_{1,1}, \ldots, kw^*_{1,N})$ for some N, and the challenger returns the challenge trapdoors of $kw^*_{b,i}$ for $i \in [1, N]$. Unfortunately, our IND-IKGA definition does not directly imply MTI. That is, if $\mathcal{A}$ is allowed to send either (kw^*_0, i^*) or (kw^*_1, i^*) to the trapdoor oracle, two trapdoors may be linked, i.e., whether these trapdoors are for the same keyword can be checked. As mentioned by Qin et al. [29], a trapdoor generation algorithm must be probabilistic to provide MTI. Because Benhamouda et al. [6] introduced a probabilistic rounding function, our construction could provide MTI. However, adaptive smoothness guarantees that when $(\mathsf{lpar}, \chi, \mathsf{hp})$ is fixed, $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{lpar}, f(\mathsf{hp}))$ is statistically close to uniform over $\{0,1\}^\nu$, but it does not directly guarantee unlinkability of two hash values, i.e., information whether two hash values are computed by the same input or not might be leaked. We leave how to provide MTI as a future work.

4 Analysis of Liu et al. Generic Construction

In this section, we analyze the Liu et al. generic construction [24] from the viewpoint of security, security models, efficiency, and instantiability of the generic construction.

4.1 Core Idea of Liu et al. Construction

First, we revisit the Liu et al. construction methodology as follows. Basically, they employed the Katz-Vaikuntanathan password-based authenticated key exchange (PAKE) construction [21] which is explained as follows. Let $\mathsf{pk}_{\mathsf{PKE}}$ be a common public key where no one knows the corresponding $\mathsf{dk}_{\mathsf{PKE}}$. A client (a sender in the PAEKS context) setups $\mathsf{hk}_c \leftarrow \mathsf{HashKG}(\mathsf{pk}_{\mathsf{PKE}})$ and $\mathsf{hp}_c \leftarrow \mathsf{ProjKG}(\mathsf{hk}_c, \mathsf{pk}_{\mathsf{PKE}})$, and a server (a receiver in the PAEKS context) setups $\mathsf{hk}_s \leftarrow \mathsf{HashKG}(\mathsf{pk}_{\mathsf{PKE}})$ and $\mathsf{hp}_s \leftarrow \mathsf{ProjKG}(\mathsf{hk}_s, \mathsf{pk}_{\mathsf{PKE}})$, respectively. Here, hp_c and hp_s are public keys and hk_c and hk_s are secret keys, respectively. The client generates an encryption C of the password pw using $\mathsf{pk}_{\mathsf{PKE}}$, and the server also generates an encryption C' of the password pw using $\mathsf{pk}_{\mathsf{PKE}}$. Then, the shared key is $\mathsf{shard\text{-}key} := \mathsf{Hash}(\mathsf{hk}_c, \mathsf{pk}_{\mathsf{PKE}}, C') \oplus \mathsf{ProjHash}(\mathsf{hp}_s, \mathsf{pk}_{\mathsf{PKE}}, C, pw) = \mathsf{Hash}(\mathsf{hk}_s, \mathsf{pk}_{\mathsf{PKE}}, C) \oplus \mathsf{ProjHash}(\mathsf{hp}_c, \mathsf{pk}_{\mathsf{PKE}}, C', pw)$. The left-side of the equation can be computed by the client using its secret key hk_c and the witness pw, and the right-side of the equation can be computed by the server using its secret key hk_s and the witness pw. The equation holds owing to the correctness of SPHF. Moreover, thanks to the word-independency, projected keys, hp_c and hp_s, can be public keys before seeing words (ciphertexts).

The basic idea of the Liu et al. PAEKS construction is to prepare an extended keyword $\mathsf{der}\text{-}kw$ from a keyword kw by $\mathsf{der}\text{-}kw \leftarrow \mathsf{HF}(kw, \mathsf{shard\text{-}key})$ where HF is a secure hash function (we intentionally use "secure hash function" in accordance with the Liu et al. description). Then, a ciphertext of $\mathsf{der}\text{-}kw$ is computed by the $\mathsf{PEKS.Enc}$ algorithm, and a trapdoor of $\mathsf{der}\text{-}kw$ is computed by the $\mathsf{PEKS.Trapdoor}$ algorithm.

4.2 Issues of the Liu et al. Construction

Security. Here, we highlight two issues regarding the underlying hash function to derive an extended keyword $\mathsf{der}\text{-}kw \leftarrow \mathsf{HF}(kw, \mathsf{shard\text{-}key})$. We note that they claimed that their construction is secure in the standard model. That is, HF is not modeled as a random oracle. Liu et al. required that if $kw \neq kw'$, then $\mathsf{der}\text{-}kw \neq \mathsf{der}\text{-}kw'$. Thus, they implicitly assumed that HF is collision resistant. However, there "exists" kw and kw' such that $kw \neq kw'$ and $\mathsf{der}\text{-}kw = \mathsf{der}\text{-}kw'$. Of course we can assume that no PPT adversary can find them, but obviously the construction does not provide statistical consistency. The first issue can be easily fixed by introducing computational consistency. However, the second issue is more important. Since a PAEKS trapdoor for kw is a PEKS trapdoor for $\mathsf{der}\text{-}kw$, information of $\mathsf{der}\text{-}kw$ is leaked from the trapdoor (because PEKS does not provide trapdoor privacy). They required that if $\mathsf{shard\text{-}key}$ is random, then $\mathsf{der}\text{-}kw$ is also random, and assumed that a hash value does not leak input information. However, since HF is not modeled as a random oracle, this claim does not hold even if HF provides one-wayness. These issues seem to be fixed by assuming that HF is a random oracle.

Regarding a hash function, we highlight that their setup algorithm is also problematic. The setup algorithm runs $(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, chooses a plaintext $M_{\mathsf{PKE}} \in \mathcal{MS}$, and generates a common public key $\mathsf{mpk} \leftarrow \mathsf{HF}(\mathsf{pk}_{\mathsf{PKE}}, M_{\mathsf{PKE}})$ where HF is a secure hash function (here we omit label from the input). Then, mpk is set as lpar of the underlying SPHF, and it is required that no one knows the corresponding decryption key. First, since $\mathsf{Setup.lpar} = \mathsf{PKE.KeyGen}$, it is not directly guaranteed that mpk works as lpar. Even if mpk is identical from a public key generated by the $\mathsf{PKE.KeyGen}$ algorithm, second, how to switch a ciphertext (a word in the SPHF context) from $\widetilde{\mathscr{L}}_{\mathsf{lpar}}$ to $\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$ in the security proof is unclear. This step is mandatory to utilize smoothness.[6] Owing to the IND-CPA security of PKE, this can be switched (as in our security proof); however, to do so, mpk must be generated by the challenger of the PKE scheme and set as the common public key. Since mpk is a hash value in the Liu et al. construction, how to set mpk as the hash value must be considered. This issue can be solved by assuming that HF is a random oracle (i.e., using the programmability of the random oracle). Alternatively, specifying $\mathsf{pk}_{\mathsf{PKE}}$ as a common public key is sufficient (even though the corresponding decryption key needs to be erased, which requires a trusted setup).

Security Model. In the definition of consistency (besides statistical or computational), they did not consider the case "$kw = kw'$ and senders are different", which is considered in our consistency definition. Since a trapdoor is associated with not only a keyword but also a sender, considering this case is important. In addition, regarding IND-CKA and IND-IKGA, they used a weak model, where an adversary $\mathcal{A}$ is not allowed to send challenge keywords to two oracles: an encryption oracle $\mathcal{O}_C$ and a trapdoor oracle $\mathcal{O}_T$.

Liu et al. also claimed that their construction provides multi-trapdoor indistinguishability (MTI) if the trapdoor algorithm of the underlying PEKS scheme is probabilistic (Theorem 5.3. [24]). Besides the probabilistic algorithm, they implicitly assumed that two trapdoors are unlinkable, i.e., it hides whether two trapdoors are generated for the same keyword or not. Then, for simulating MTI game in the proof of Theorem 3.3. [24], a simulator just responses random values as the challenge trapdoors. However, even if the trapdoor algorithm is probabilistic, it does not directly guarantee the unlinkability.

Efficiency. They employed two SPHFs as in the Katz-Vaikuntanathan PAKE construction. In PAKE, this is mandatory because both a client and a server need to show the possession of witness (password). However, this is not the case in PAEKS: a receiver does not have to prove the possession of witness. In our construction, a sender is required to show the possession of witness (randomness

[6]Precisely, Liu et al. did not directly employ smoothness. They assumed that a hash value H of SPHF for a word $\chi \in \widetilde{\mathscr{L}}_{\mathsf{lpar}}$ is random (they called it pseudo-randomness), and they switched H to be random without switching the word from $\widetilde{\mathscr{L}}_{\mathsf{lpar}}$ to $\mathcal{X}_{\mathsf{lpar}} \setminus \mathscr{L}_{\mathsf{lpar}}$. Even if this argument is correct, still $\mathsf{mpk} = \mathsf{lpar}$ needs to be set as a hash value $\mathsf{mpk} \leftarrow \mathsf{HF}(\mathsf{pk}_{\mathsf{PKE}}, M_{\mathsf{PKE}})$.

of a ciphertext), and a receiver just uses a hash key for computing the hash value. This improves the efficiency of the construction.

Instantiability. Finally, we discuss the instantiability of the Liu et al. generic construction (besides hash functions discussed above). They employed a labeled IND-CCA2 PKE scheme that defines languages of ciphertexts of SPHF. However, Benhamouda et al. [6] proposed a word-independent SPHF for a IND-CPA PKE scheme, and mentioned that their SPHF construction for the IND-CCA2 PKE scheme is not word-independent. Moreover, Liu et al. employed an IND-CCA1 PKE scheme in their implementation. Although the Liu et al. construction may work well since IND-CCA2 implies IND-CPA, however, it is not clear whether the underlying SPHF is word-independent (at least in their implementation). In the next section, we show that the underlying PKE is required to be IND-CPA secure, and no CCA2/CCA1 security is required.

We also remark that one of the main goals of Liu et al. was to construct a post-quantum PAEKS scheme by instantiating their generic construction from lattices. Thus, even if the above-mentioned issues regarding hash functions can be solved by introducing random oracles, it would be better to determine whether these issues can be solved, even in the quantum random oracle model.

5 Our PAEKS Construction

In this section, we propose a generic construction of PAEKS.

5.1 Proposed Generic Construction

High-level Description. We also employ the methodology of Katz-Vaikuntanathan PAKE construction. As an important difference from the Liu et al. construction, we introduce one SPHF where a receiver publishes a projected key hp, and a sender generates a ciphertext $\mathsf{ct}_{\mathsf{PKE}}$ of 0 and sets it as a public key $\mathsf{pk}_{\mathsf{S}} = \mathsf{ct}_{\mathsf{PKE}}$. Its randomness ρ_S is set as a secret key sk_{S}. For running the $\mathsf{ProjHash}$ algorithm in the $\mathsf{PAEKS.Enc}$ algorithm, $\mathsf{sk}_{\mathsf{S}} = \rho_S$ is used as the witness. That is, for generating a ciphertext of a keyword, a secret key $\mathsf{sk}_{\mathsf{S}} = \rho_S$ is required, whereas for generating a trapdoor, a public key $\mathsf{pk}_{\mathsf{S}} = \mathsf{ct}_{\mathsf{PKE}}$ is enough in addition to sk_{R}. This setting matches the syntax of PAEKS. Since a word $\mathsf{ct}_{\mathsf{PKE}}$ is generated after seeing hp, the underlying SPHF needs to be word-independent. Here, the public key of PKE $\mathsf{pk}_{\mathsf{PKE}}$ is generated by the receiver, and the sender needs to know $\mathsf{pk}_{\mathsf{PKE}}$ for running the $\mathsf{PAEKS.KG}_{\mathsf{S}}$ algorithm. Thus, our construction employs a designated-receiver setting. For deriving an extended keyword $\mathsf{der}\text{-}kw$, we employ a PRF where a PRF key is a hash value of SPHF. Intuitively, after utilizing adaptive smoothness, a hash value of SPHF is random. Then, owing to the pseudo-randomness of PRF, $\mathsf{der}\text{-}kw$ is also random. For extended keywords, we employ a PEKS scheme. As in the usual PEKS setting, the receiver runs $(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$, and $\mathsf{sk}_{\mathsf{PEKS}}$ is used for generating a trapdoor for $\mathsf{der}\text{-}kw$.

Let $\mathsf{PRF} = \{F_K : \mathcal{KS} \to \mathcal{KS}\}$ be a family of PRFs where $K \in \{0,1\}^\nu$, $\mathsf{PEKS} = (\mathsf{PEKS.KG}, \mathsf{PEKS.Enc}, \mathsf{PEKS.Trapdoor}, \mathsf{PEKS.Test})$ be a PEKS scheme with a keyword space $\mathcal{KS}$, $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme, and $\mathsf{WI\text{-}SPHF} = (\mathsf{HashKG}, \mathsf{ProjKG}, \mathsf{Hash}, \mathsf{ProjHash})$ be a word-independent SPHF with the output length $\{0,1\}^\nu$. We remark that, because we employ a hash value of SPHF as a PRF key, we assume that $K \in \{0,1\}^\nu$, and because a PRF takes a keyword as input and outputs an extended keyword, we assume that $F_K : \mathcal{KS} \to \mathcal{KS}$. We also assume that $2^{-\nu}$ is negligible in the security parameter λ to guarantee that for randomly chosen two values $\mathsf{H}, \mathsf{H}' \xleftarrow{\$} \{0,1\}^\nu$, $\mathsf{H} \neq \mathsf{H}'$ holds with overwhelming probability.

Proposed Generic Construction

$\mathsf{PAEKS.KG_R}(1^\lambda)$**:** Run $(\mathsf{pk_{PKE}}, \mathsf{dk_{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pk_{PKE}})$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{pk_{PKE}})$, and $(\mathsf{pk_{PEKS}}, \mathsf{sk_{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$. Set $\mathsf{pk_R} = (\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{pk_{PEKS}})$ and $\mathsf{sk_R} = (\mathsf{hk}, \mathsf{dk_{PKE}}, \mathsf{sk_{PEKS}})$, and output $(\mathsf{pk_R}, \mathsf{sk_R})$.

$\mathsf{PAEKS.KG_S}(\mathsf{pk_R})$**:** Parse $\mathsf{pk_R} = (\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{pk_{PEKS}})$. Run $\mathsf{ct_{PKE}} = \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, 0; \rho_S)$, set $\mathsf{pk_S} = \mathsf{ct_{PKE}}$ and $\mathsf{sk_S} = \rho_S$, and output $(\mathsf{pk_S}, sks)$. If we explicitly describe the sender index i, then we denote $\mathsf{pk_{S[\mathit{i}]}} = \mathsf{ct}^{(i)}_{\mathsf{PKE}}$ and $\mathsf{sk_{S[\mathit{i}]}} = \rho_S^{(i)}$.

$\mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, kw)$**:** Parse $\mathsf{pk_R} = (\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{pk_{PEKS}})$, $\mathsf{pk_S} = \mathsf{ct_{PKE}}$, and $\mathsf{sk_S} = \rho_S$. Compute $\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{ct_{PKE}}, \rho_S)$ and run $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{pH}}(kw)$. Compute $\mathsf{ct_{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk_{PEKS}}, \mathsf{der}\text{-}kw)$, set $\mathsf{ct_{PAEKS}} = \mathsf{ct_{PEKS}}$, and output $\mathsf{ct_{PAEKS}}$.

$\mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw)$**:** Parse $\mathsf{pk_R} = (\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{pk_{PEKS}})$, $\mathsf{pk_S} = \mathsf{ct_{PKE}}$, and $\mathsf{sk_R} = (\mathsf{hk}, \mathsf{dk_{PKE}}, \mathsf{sk_{PEKS}})$. Compute $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{pk_{PKE}}, \mathsf{ct_{PKE}})$ and run $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{H}}(kw)$. Compute $\mathsf{td}_{\mathsf{der}\text{-}kw} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk_{PEKS}}, \mathsf{sk_{PEKS}}, \mathsf{der}\text{-}kw)$, set $\mathsf{td}_{\mathsf{S},kw} = \mathsf{td}_{\mathsf{der}\text{-}kw}$, and output $\mathsf{td}_{\mathsf{S},kw}$.

$\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}_{\mathsf{S},kw})$**:** Parse $\mathsf{ct_{PAEKS}} = \mathsf{ct_{PEKS}}$ and $\mathsf{td}_{\mathsf{S},kw} = \mathsf{td}_{\mathsf{der}\text{-}kw}$. Output the result of $\mathsf{PEKS.Test}(\mathsf{ct_{PEKS}}, \mathsf{td}_{kw})$.

If PKE is correct, then $\mathsf{ct_{PKE}} \in \widetilde{\mathscr{L}}_{\mathsf{pk_{PKE}}}$ where $\mathsf{ct_{PKE}} = \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, 0; \rho_S)$. Moreover, if $\mathsf{WI\text{-}SPHF}$ is correct, then $\mathsf{pH} = \mathsf{H}$ holds with overwhelming probability where $\mathsf{pH} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{ct_{PKE}}, \rho_S)$ and $\mathsf{H} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{pk_{PKE}}, \mathsf{ct_{PKE}})$. Moreover, if PEKS is correct, then $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}_{\mathsf{S},kw}) = 1$ where $\mathsf{ct_{PAEKS}} = \mathsf{ct_{PEKS}}$. Thus, the proposed construction provides correctness if these building blocks provide correctness.

Feasibility of our Generic Construction. In our construction, we employ $\mathsf{WI\text{-}SPHF}$, PKE, PRF, and PEKS as building blocks. Any PRF can be employed, e.g., HMAC-SHA-256. Since Benhamouda et al. [6] proposed a $\mathsf{WI\text{-}SPHF}$ for a lattice-based IND-CPA PKE scheme, our construction is feasible. Since PEKS can be generically constructed from anonymous IBE [1], and lattice-based IBE is basically anonymous, our generic construction yields post-quantum PAEKS. Behnia et al. [4] gave implementations of lattice-based PEKS schemes when Agrawal-Boneh-Boyen IBE [2] or Ducas-Lyubashevsky-Prest IBE [15] are employed as the underlying anonymous IBE scheme. These PEKS schemes may be attractive in terms of efficiency.

Non Designated-Receiver Setting. As previously mentioned, if we assume a trusted setup, then we do not have to introduce the designated-receiver setting. Concretely, a setup algorithm takes a security parameter λ as input, runs $(\mathsf{pk_{PKE}}, \mathsf{dk_{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, erases $\mathsf{dk_{PKE}}$, and outputs $\mathsf{pk_{PKE}}$ only as a public parameter. However, it would be better to clarify who can know $\mathsf{dk_{PKE}}$ since $\mathsf{dk_{PKE}} = \mathsf{ltrap}$ can be used for breaking the underlying membership problem. In this sense, our setting is more desirable.

5.2 Security Analysis

Theorem 1. *Our construction is computationally consistent if* PEKS *is computationally consistent,* PKE *is IND-CPA secure,* $\mathsf{WI\text{-}SPHF}$ *provides correctness and adaptive smoothness, and* PRF *is pseudo-random.*

Proof Overview. Since the underlying PEKS scheme PEKS is computationally consistent, what we need to show is as follows: for $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{pH}^{(i)}}(kw)$ (computed in the $\mathsf{PAEKS.Enc}$ algorithm in the experiment) and $\mathsf{der}\text{-}kw' \leftarrow F_{\mathsf{H}^{(j)}}(kw')$ (computed in the $\mathsf{PAEKS.Trapdoor}$ algorithm in the experiment), $\mathsf{der}\text{-}kw \neq \mathsf{der}\text{-}kw'$ holds if $(kw, i) \neq (kw', j)$, where $\mathsf{pH}^{(i)} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{ct}^{(i)}_{\mathsf{PKE}}, \rho^{(i)}_S)$ and $\mathsf{H}^{(j)} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{pk_{PKE}}, \mathsf{ct}^{(j)}_{\mathsf{PKE}})$. Our proof strategy is explained as follows. First, the computation of $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{pH}^{(i)}}(kw)$ is switched to $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{H}^{(i)}}(kw)$ where $\mathsf{H}^{(i)} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{pk_{PKE}}, \mathsf{ct}^{(i)}_{\mathsf{PKE}})$. Owing to the correctness of WI-SPHF, this modification is indistinguishable. Next, $\mathsf{pk}_{\mathsf{S}[i]} = \mathsf{ct}^{(i)}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, 0; \rho^{(i)}_S)$ computed in the $\mathsf{PAEKS.KG_S}$ algorithm is switched to $\mathsf{ct}^{(i)}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, 1; \rho^{(i)}_S)$. In the SPHF-context, a word $\mathsf{ct}^{(i)}_{\mathsf{PKE}} \in \widetilde{\mathscr{L}}_{\mathsf{pk_{PKE}}}$ is switched to $\mathsf{ct}^{(i)}_{\mathsf{PKE}} \in \mathcal{X}_{\mathsf{pk_{PKE}}} \setminus \mathscr{L}_{\mathsf{pk_{PKE}}}$. Owing to the IND-CPA security of PKE, this modification is indistinguishable. Then we can utilize adaptive smoothness, i.e., for computing $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{H}^{(i)}}(kw)$, $\mathsf{H}^{(i)}$ is randomly chosen from $\{0,1\}^\nu$. $\mathsf{pk}_{\mathsf{S}[j]} = \mathsf{ct}^{(j)}_{\mathsf{PKE}}$ is also switched as well, and for $\mathsf{der}\text{-}kw' \leftarrow F_{\mathsf{H}^{(j)}}(kw')$, $\mathsf{H}^{(j)}$ is randomly chosen from $\{0,1\}^\nu$ (if $i = j$, then two values are the same). Then, PRF keys $\mathsf{H}^{(i)}$ and $\mathsf{H}^{(j)}$ are random. Thus, we can replace $F_{\mathsf{H}^{(i)}}$ and $F_{\mathsf{H}^{(j)}}$ with random functions owing to the pseudo-randomness of PRF. If $i = j$, then $kw \neq kw'$. Then, $\mathsf{der}\text{-}kw$ and $\mathsf{der}\text{-}kw'$ are randomly chosen, respectively, and $\mathsf{der}\text{-}kw \neq \mathsf{der}\text{-}kw'$ holds with overwhelming probability. If $kw = kw'$, then $i \neq j$. Then, $F_{\mathsf{H}^{(i)}}$ and $F_{\mathsf{H}^{(j)}}$ are different random functions. Thus $\mathsf{der}\text{-}kw$ and $\mathsf{der}\text{-}kw'$ are randomly chosen, respectively (even the input is the same), and $\mathsf{der}\text{-}kw \neq \mathsf{der}\text{-}kw'$ holds with overwhelming probability. Finally, owing to the computational consistency of PEKS, our construction provides computational consistency.

Proof. We prove the theorem via sequences of games $\mathsf{Game}_0, \ldots, \mathsf{Game}_8$. Let W_k denote an event that $\mathcal{A}$ wins in Game_k ($k \in \{0, 1, \ldots, 8\}$). Without loss of generality, $(i, j) \in \{0,1\} \times \{0,1\}$ is determined before generating $(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[0]}, \mathsf{pk}_{\mathsf{S}[1]})$ with the probability $1/4$.

Game_0**:** This game is the same as the original computational consistency game in Definition 14.

Game_1**:** This game is the same as Game_0 except that the computation of $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}, kw)$ is changed as follows. Let $\mathsf{sk_R} = (\mathsf{hk}, \mathsf{dk_{PKE}}, \mathsf{sk_{PEKS}})$. Instead of computing $\mathsf{pH}^{(i)} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{ct}^{(i)}_{\mathsf{PKE}}, \rho^{(i)}_S)$, compute $\mathsf{H}^{(i)} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{pk_{PKE}}, \mathsf{ct}^{(i)}_{\mathsf{PKE}})$ and run $\mathsf{der}\text{-}kw \leftarrow F_{\mathsf{H}^{(i)}}(kw)$. $\mathcal{B}$ computes $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk_{PEKS}}, \mathsf{der}\text{-}kw)$. If WI-SPHF provides correctness, then $\mathsf{pH}^{(i)} = \mathsf{H}^{(i)}$ holds with overwhelming probability. Thus, $|\Pr[W_0] - \Pr[W_1]|$ is negligible in the security parameter λ.

Game_2**:** This game is the same as Game_1 except that $(\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})$ is changed as $\mathsf{pk}_{\mathsf{S}[i]} = \mathsf{ct}^{(i)}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, 1; \rho^{(i)}_S)$. We show that $|\Pr[W_1] - \Pr[W_2]|$ is negligible if PKE is IND-CPA secure as follows. Let $\mathcal{C}$ be the challenger of the IND-CPA game. We construct an algorithm $\mathcal{B}$ as follows. $\mathcal{C}$ runs $(\mathsf{pk_{PKE}}, \mathsf{dk_{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and sends $\mathsf{pk_{PKE}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pk_{PKE}})$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{pk_{PKE}})$, and $(\mathsf{pk_{PEKS}}, \mathsf{sk_{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$, and sets $\mathsf{pk_R} = (\mathsf{hp}, \mathsf{pk_{PKE}}, \mathsf{pk_{PEKS}})$. For running $(\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})$, $\mathcal{B}$ sends $(M^*_0, M^*_1) = (0, 1)$ to $\mathcal{C}$. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0,1\}$, computes $\mathsf{ct}^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, M^*_b)$, and sends $\mathsf{ct}^*_{\mathsf{PKE}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $(\mathsf{pk}_{\mathsf{S}[\bar{i}]}, \mathsf{sk}_{\mathsf{S}[\bar{i}]}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})$ as usual, where $\bar{i} = 0$ if $i = 1$ and $\bar{i} = 1$ if $i = 0$, and sends $(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[0]}, \mathsf{pk}_{\mathsf{S}[1]})$ to $\mathcal{A}$. We note that $\rho^{(i)}_S$ is not used for running the $\mathsf{PAEKS.Enc}$ algorithm due to the modification in Game_1. If $b = 0$, then $\mathcal{B}$ simulates Game_1 and if $b = 1$, $\mathcal{B}$ simulates Game_2. Thus, $|\Pr[W_1] - \Pr[W_2]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$ holds.

Game_3**:** This game is the same as Game_2 except that $\mathsf{H}^{(i)} \xleftarrow{\$} \{0,1\}^\nu$. Since $\mathsf{ct}^{(i)}_{\mathsf{PKE}} \in \mathcal{X}_{\mathsf{pk_{PKE}}} \setminus \mathscr{L}_{\mathsf{pk_{PKE}}}$, $|\Pr[W_2] - \Pr[W_3]|$ is negligible if WI-SPHF provides adaptive smoothness.

Game_4**:** This game is the same as Game_3 except that $(\mathsf{pk}_{\mathsf{S}[j]}, \mathsf{sk}_{\mathsf{S}[j]}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pk_R})$ is changed as $\mathsf{pk}_{\mathsf{S}[j]} = \mathsf{ct}^{(j)}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{PKE}}, 1; \rho^{(j)}_S)$. As in Game_2, $|\Pr[W_3] - \Pr[W_4]|$ is negligible if PKE is IND-CPA secure. If $i = j$, then skip this game.

Game_5**:** This game is the same as Game_4 except that $\mathsf{H}^{(j)} \xleftarrow{\$} \{0,1\}^\nu$. As in Game_4, $|\Pr[W_4]-\Pr[W_5]|$ is negligible if $\mathsf{WI\text{-}SPHF}$ provides adaptive smoothness. If $i = j$, then skip this game.

Game_6**:** This game is the same as Game_5 except that $\mathsf{der}\text{-}kw \xleftarrow{\$} \mathcal{KS}$. We show that $|\Pr[W_5]-\Pr[W_6]|$ is negligible if PRF is pseudo-random as follows. Let $\mathcal{C}$ be the challenger of PRF. We construct an algorithm $\mathcal{B}$ as follows. $\mathcal{B}$ prepares all keys as in Game_5. When $\mathcal{B}$ prepares $\mathsf{der}\text{-}kw$, $\mathcal{B}$ sends kw to $\mathcal{C}$. Then, $\mathcal{C}$ returns either $F_K(kw)$ or a random R. In the former case, $\mathcal{B}$ simulates Game_5 (implicitly set $K = \mathsf{H}^{(i)}$), and in the latter case, $\mathcal{B}$ simulates Game_6. Thus, $|\Pr[W_5] - \Pr[W_6]| \leq \mathsf{Adv}^{\mathsf{pseudo\text{-}random}}_{\mathsf{PRF},\mathcal{B}}(\lambda)$ holds.

Game_7**:** This game is the same as Game_6 except that $\mathsf{der}\text{-}kw' \xleftarrow{\$} \mathcal{KS}$. As in Game_6, $|\Pr[W_6]-\Pr[W_7]|$ is negligible if PRF is pseudo-random.

Game_8**:** We show that the probability that $\mathcal{A}$ wins in this game is negligible as follows. Let $\mathcal{C}$ be the challenger of computational consistency of PEKS. We construct an algorithm $\mathcal{B}$ as follows. $\mathcal{C}$ runs $(\mathsf{pk_{PEKS}}, \mathsf{sk_{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$ and sends $\mathsf{pk_{PEKS}}$ to $\mathcal{B}$. $\mathcal{B}$ prepares $(\mathsf{pk_R}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{pk}_{\mathsf{S}[j]})$ in accordance with the previous game, except that $\mathcal{B}$ contains $\mathsf{pk_{PEKS}}$ to $\mathsf{pk_R}$. When $\mathcal{A}$ sends $(kw, kw', i.j)$ to $\mathcal{B}$, $\mathcal{B}$ randomly chooses $\mathsf{der}\text{-}kw, \mathsf{der}\text{-}kw' \xleftarrow{\$} \mathcal{KS}$, and sends $(\mathsf{der}\text{-}kw, \mathsf{der}\text{-}kw')$ to $\mathcal{C}$. Here, the probability of $\mathsf{der}\text{-}kw = \mathsf{der}\text{-}kw'$ is $2^{-\nu}$, and is negligible. $\mathcal{C}$ runs $\mathsf{ct_{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk_{PEKS}}, \mathsf{der}\text{-}kw)$ and $\mathsf{td}_{\mathsf{der}\text{-}kw'} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk_{PEKS}}, \mathsf{sk_{PEKS}}, \mathsf{der}\text{-}kw')$. Since PEKS provides computational consistency, the probability that $\mathsf{PEKS.Test}(\mathsf{ct_{PEKS}}, \mathsf{td}_{\mathsf{der}\text{-}kw'}) = 1$ holds is negligible. This concludes the proof. □

Theorem 2. *Our construction is IND-CKA secure if* PEKS *is IND-CKA secure,* PKE *is IND-CPA secure,* $\mathsf{WI\text{-}SPHF}$ *provides correctness and adaptive smoothness, and* PRF *is pseudo-random.*

Proof Overview. Basically, the theorem holds if PEKS is IND-CKA secure since a PAEKS ciphertext is a PEKS ciphertext in our construction. What we need to care is: $\mathcal{O}_T$ in the PAEKS experiment needs to be simulated by $\mathcal{O}_T$ in the PEKS experiment. We remark that if $\mathcal{A}$ sends kw to $\mathcal{O}_T$ in the PAEKS experiment, the keyword to be input to $\mathcal{O}_T$ in the PEKS experiment is not kw, is $\mathsf{der}\text{-}kw$. That is, if there exists $kw \notin \{kw^*_0, kw^*_1\}$ such that its extended keyword $\mathsf{der}\text{-}kw \in \{\mathsf{der}\text{-}kw^*_0, \mathsf{der}\text{-}kw^*_1\}$ where $\mathsf{der}\text{-}kw^*_0$ and $\mathsf{der}\text{-}kw^*_1$ are extended keywords of kw^*_0 and kw^*_1, respectively, then the simulation fails (because the challenge ciphertext is computed by $\mathsf{der}\text{-}kw^*_b$ for $b \xleftarrow{\$} \{0,1\}$). To exclude this case, we show that all extended keywords are random using the same strategy of the proof of computational consistency. First, $\mathsf{pH}^{(i)}$ is switched to $\mathsf{H}^{(i)}$ owing to the correctness of $\mathsf{WI\text{-}SPHF}$. Second, $\mathsf{ct}^{(i)}_{\mathsf{PKE}}$ is switched to a ciphertext of 1 owing to the IND-CPA security of PKE. Third, $\mathsf{H}^{(i)}$ is switched to a random value owing to the adaptive smoothness of $\mathsf{WI\text{-}SPHF}$. For all $i \in [1, n]$, run the above procedures. Then, all extended keywords are randomly chosen because of the pseudo-randomness of PRF. Here, the maximum number of extended keywords is at most $q_C + q_T + 2$ where q_C is the number of encryption queries, q_T is the number of trapdoor queries, and 2 is for generating the challenge ciphertext, i.e,, for (kw^*_0, kw^*_1, i^*) that $\mathcal{A}$ declares, and two extended keywords $\mathsf{der}\text{-}kw^*_0$ and $\mathsf{der}\text{-}kw^*_1$ are defined. Finally, owing to the IND-CKA security of PEKS, our construction is IND-CKA secure. We remark that $\mathcal{O}_C$ in the PAEKS experiment can be simulated during extended keyword randomization phases as in the same strategy of the proof of computational consistency that also prepares a ciphertext. To simulate $\mathcal{O}_T$ during extended keyword randomization phases, the simulator runs $(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$, and uses $\mathsf{sk}_{\mathsf{PEKS}}$ to compute a trapdoor.

Proof. We prove the theorem via sequences of games $\mathsf{Game}_0, \dots, \mathsf{Game}_5$. Let W_k denote an event that $\mathcal{A}$ wins in Game_k $(k \in \{0, 1, \dots, 5\})$.

Game_0**:** This game is the same as the original IND-CKA game in Definition 15.

Next, we introduce subgames $\mathsf{Game}_{1,1}, \mathsf{Game}_{2,1}, \mathsf{Game}_{3,1}, \mathsf{Game}_{1,2}, \mathsf{Game}_{2,2}, \mathsf{Game}_{3,2}, \dots, \mathsf{Game}_{1,n}, \mathsf{Game}_{2,n}, \mathsf{Game}_{3,n}$.

$\mathsf{Game}_{1,i}$**:** Set $\mathsf{Game}_{3,0} = \mathsf{Game}_0$. We describe $\mathsf{Game}_{1,i}$ $(i \in [1, n])$ as follows. This game is the same as $\mathsf{Game}_{3,i-1}$ except that compute $\mathsf{H}^{(i)} \leftarrow \mathsf{Hash}(\mathsf{hk}, \mathsf{pk}_{\mathsf{PKE}}, \mathsf{ct}^{(i)}_{\mathsf{PKE}})$ instead of computing $\mathsf{pH}^{(i)} \leftarrow \mathsf{ProjHash}(\mathsf{hp}, \mathsf{pk}_{\mathsf{PKE}}, \mathsf{ct}^{(i)}_{\mathsf{PKE}}, \rho^{(i)}_S)$. If WI-SPHF provides correctness, then $\mathsf{pH}^{(i)} = \mathsf{H}^{(i)}$ holds with overwhelming probability. Thus, $|\Pr[W_{3,i-1}] - \Pr[W_{1,i}]|$ is negligible in the security parameter λ.

$\mathsf{Game}_{2,i}$**:** We describe $\mathsf{Game}_{2,i}$ $(i \in [1, n])$ as follows. This game is the same as $\mathsf{Game}_{1,i}$ except that $(\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}) \leftarrow \mathsf{PAEKS.KG}_{\mathsf{S}}(\mathsf{pk}_{\mathsf{R}})$ is changed as $\mathsf{pk}_{\mathsf{S}[i]} = \mathsf{ct}^{(i)}_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, 1; \rho^{(i)}_S)$. We show that $|\Pr[W_{1,i}] - \Pr[W_{2,i}]|$ is negligible if PKE is IND-CPA secure as follows. Let $\mathcal{C}$ be the challenger of the IND-CPA game. We construct an algorithm $\mathcal{B}$ as follows. $\mathcal{C}$ runs $(\mathsf{pk}_{\mathsf{PKE}}, \mathsf{dk}_{\mathsf{PKE}}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and sends $\mathsf{pk}_{\mathsf{PKE}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\mathsf{hk} \leftarrow \mathsf{HashKG}(\mathsf{pk}_{\mathsf{PKE}})$, $\mathsf{hp} \leftarrow \mathsf{ProjKG}(\mathsf{hk}, \mathsf{pk}_{\mathsf{PKE}})$, and $(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$, and sets $\mathsf{pk}_{\mathsf{R}} = (\mathsf{hp}, \mathsf{pk}_{\mathsf{PKE}}, \mathsf{pk}_{\mathsf{PEKS}})$. $\mathcal{B}$ generates $(\mathsf{pk}_{\mathsf{S}[1]}, \mathsf{sk}_{\mathsf{S}[1]}), \dots, (\mathsf{pk}_{\mathsf{S}[i-1]}, \mathsf{sk}_{\mathsf{S}[i-1]})$ as in $\mathsf{Game}_{1,i}$. For running $(\mathsf{pk}_{\mathsf{S}[i]}, \mathsf{sk}_{\mathsf{S}[i]}) \leftarrow \mathsf{PAEKS.KG}_{\mathsf{S}}(\mathsf{pk}_{\mathsf{R}})$, $\mathcal{B}$ sends $(M^*_0, M^*_1) = (0, 1)$ to $\mathcal{C}$. $\mathcal{C}$ chooses $b \xleftarrow{\$} \{0, 1\}$, computes $\mathsf{ct}^*_{\mathsf{PKE}} \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{PKE}}, M^*_b)$, and sends $\mathsf{ct}^*_{\mathsf{PKE}}$ to $\mathcal{B}$. $\mathcal{B}$ generates $(\mathsf{pk}_{\mathsf{S}[i+1]}, \mathsf{sk}_{\mathsf{S}[i+1]}), \dots, (\mathsf{pk}_{\mathsf{S}[n]}, \mathsf{sk}_{\mathsf{S}[n]})$ as usual and sends $\mathsf{pk}_{\mathsf{R}}, \{\mathsf{pk}_{\mathsf{S}[i]}\}_{i \in [1,n]}$ to $\mathcal{A}$. For an encryption query (kw, i), B uses $\mathsf{H}^{(i)}$ and it does not require $\rho^{(i)}_S$. So, even if $\mathsf{pk}_{\mathsf{S}[i]} = \mathsf{ct}^{(i)}_{\mathsf{PKE}}$ is switched to a ciphertext of 1, $\mathcal{B}$ can answer the query. For a trapdoor query (kw, i), B generates a trapdoor $\mathsf{td}_{\mathsf{S}[i],kw}$ using $\mathsf{sk}_{\mathsf{PEKS}}$. If $b = 0$, then $\mathcal{B}$ simulates $\mathsf{Game}_{1,i}$ and if $b = 1$, $\mathcal{B}$ simulates $\mathsf{Game}_{2,i}$. Thus, $|\Pr[W_{1,i}] - \Pr[W_{2,i}]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$ holds.

$\mathsf{Game}_{3,i}$**:** We describe $\mathsf{Game}_{3,i}$ $(i \in [1, n])$ as follows. This game is the same as $\mathsf{Game}_{2,i}$ except that $\mathsf{H}^{(i)} \xleftarrow{\$} \{0, 1\}^\nu$. $|\Pr[W_{2,i}] - \Pr[W_{3,i}]|$ is negligible if WI-SPHF provides adaptive smoothness.

Next, we introduce subgames $\mathsf{Game}_{4,0}, \dots, \mathsf{Game}_{4,N}$ where $N \leq q_C + q_T + 2$ is the number of extended keywords appeared in the game. Here, q_C is the number of encryption queries and q_T is the number of trapdoor queries.

$\mathsf{Game}_{4,i}$**:** Set $\mathsf{Game}_{4,0} = \mathsf{Game}_{3,n}$. Let $\{\mathsf{der}\text{-}kw_1, \dots, \mathsf{der}\text{-}kw_N\}$ be the set of distinct extended keywords. This game is the same as $\mathsf{Game}_{4,i-1}$ except that $\mathsf{der}\text{-}kw_i \xleftarrow{\$} \mathcal{KS}$. We show that $|\Pr[W_{4,i-1}] - \Pr[W_{4,i}]|$ is negligible if PRF is pseudo-random. Let $\mathcal{C}$ be the challenger of PRF. We construct an algorithm $\mathcal{B}$ as follows. When $\mathcal{B}$ prepares $\mathsf{der}\text{-}kw_i$ for the i-th query (kw, j)[7] for some $j \in [1, n]$, $\mathcal{B}$ sends kw to $\mathcal{C}$. Then, $\mathcal{C}$ returns either $F_K(kw)$ or a random value. In the former case, $\mathcal{B}$ simulates $\mathsf{Game}_{4,i-1}$ (implicitly set $K = \mathsf{H}^{(j)}$), and in the latter case, $\mathcal{B}$ simulates $\mathsf{Game}_{4,i}$. Thus, $|\Pr[W_{4,i-1}] - \Pr[W_{4,i}]| \leq \mathsf{Adv}^{\mathsf{pseudo\text{-}random}}_{\mathsf{PRF},\mathcal{B}}(\lambda)$ holds.

[7]It may be an encryption query, a trapdoor query, or the challenge query. In the challenge query, $j = i^*$.

Game_5**:** Set $\mathsf{Game}_{4,N} = \mathsf{Game}_5$. We show that the probability that $\mathcal{A}$ wins in this game is negligible if PEKS is IND-CKA secure. Let $\mathcal{C}$ be the challenger of the IND-CKA game of PEKS. We construct an algorithm $\mathcal{B}$ as follows. $\mathcal{C}$ runs $(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}) \leftarrow \mathsf{PEKS.KG}(1^\lambda)$ and sends $\mathsf{pk}_{\mathsf{PEKS}}$ to $\mathcal{B}$. $\mathcal{B}$ prepares $(\mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}[i]}, \mathsf{pk}_{\mathsf{S}[j]})$ in accordance with the previous game, except that $\mathcal{B}$ contains $\mathsf{pk}_{\mathsf{PEKS}}$ to pk_{R}. When $\mathcal{A}$ sends an encryption query (kw, i) to $\mathcal{B}$, if (kw, i) has been queried as either an encryption query or a trapdoor query, then $\mathcal{B}$ uses $\mathsf{der}\text{-}kw$ that was previously generated. Otherwise, $\mathcal{B}$ randomly chooses $\mathsf{der}\text{-}kw \xleftarrow{\$} \mathcal{KS}$, preserves $(\mathsf{der}\text{-}kw, kw, i)$, runs $\mathsf{ct}_{\mathsf{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{der}\text{-}kw)$, and returns $\mathsf{ct}_{\mathsf{PEKS}}$ to $\mathcal{A}$. When $\mathcal{A}$ sends a trapdoor query (kw, i) to $\mathcal{B}$, if (kw, i) has been queried as either an encryption query or a trapdoor query, then $\mathcal{B}$ uses $\mathsf{der}\text{-}kw$ that was previously generated. Otherwise, $\mathcal{B}$ randomly chooses $\mathsf{der}\text{-}kw \xleftarrow{\$} \mathcal{KS}$, preserves $(\mathsf{der}\text{-}kw, kw, i)$, and sends $\mathsf{der}\text{-}kw$ to $\mathcal{C}$ as a trapdoor query. $\mathcal{C}$ runs $\mathsf{td}_{\mathsf{der}\text{-}kw} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, \mathsf{der}\text{-}kw)$ and sends $\mathsf{td}_{\mathsf{der}\text{-}kw}$ to $\mathcal{B}$. $\mathcal{B}$ returns $\mathsf{td}_{\mathsf{der}\text{-}kw}$ to $\mathcal{A}$. In the challenge phase, $\mathcal{A}$ sends (kw_0^*, kw_1^*, i^*) to $\mathcal{B}$. If (kw_0^*, i^*) (resp. (kw_1^*, i^*)) has appeared in an encryption query, then $\mathcal{B}$ uses $\mathsf{der}\text{-}kw_0^*$ (resp. $\mathsf{der}\text{-}kw_1^*$) that was previously generated. Otherwise, $\mathcal{B}$ randomly chooses $\mathsf{der}\text{-}kw_0^* \xleftarrow{\$} \mathcal{KS}$ (resp. $\mathsf{der}\text{-}kw_1^* \xleftarrow{\$} \mathcal{KS}$). $\mathcal{B}$ sends $(\mathsf{der}\text{-}kw_0^*, \mathsf{der}\text{-}kw_1^*)$ to $\mathcal{C}$ as the challenge query. We remark that the probability that either $\mathsf{der}\text{-}kw_0^*$ or $\mathsf{der}\text{-}kw_1^*$ has been appeared as a trapdoor query is negligible since all extended keywords are randomly chosen from $\{0,1\}^\nu$. $\mathcal{C}$ randomly selects $b \xleftarrow{\$} \{0,1\}$, runs $\mathsf{ct}^*_{\mathsf{PEKS}} \leftarrow \mathsf{PEKS.Enc}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{der}\text{-}kw_b^*)$, and returns $\mathsf{ct}^*_{\mathsf{PEKS}}$ to $\mathcal{B}$. $\mathcal{B}$ returns $\mathsf{ct}^*_{\mathsf{PEKS}}$ to $\mathcal{A}$. $\mathcal{B}$ simulates $\mathcal{O}_C$ and $\mathcal{O}_T$ as in the previous stage, Finally, $\mathcal{A}$ outputs b'. $\mathcal{B}$ also outputs the same b'. Then, $\mathcal{B}$ can break IND-CKA security of PEKS with the same advantage of $\mathcal{A}$. That is, $\Pr[W_5] \leq \mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PEKS},\mathcal{A}}(\lambda, n)$ holds. This concludes the proof. $\square$

Theorem 3. *Our construction is IND-IKGA secure if* PKE *is IND-CPA secure,* $\mathsf{WI\text{-}SPHF}$ *provides correctness and adaptive smoothness, and* PRF *is pseudo-random.*

Proof Overview. The underlying PEKS scheme does not provide trapdoor privacy, i.e., from $\mathsf{td}_{\mathsf{der}\text{-}kw} \leftarrow \mathsf{PEKS.Trapdoor}(\mathsf{pk}_{\mathsf{PEKS}}, \mathsf{sk}_{\mathsf{PEKS}}, \mathsf{der}\text{-}kw)$, information of $\mathsf{der}\text{-}kw$ is leaked. So, the $\mathsf{PEKS.Trapdoor}$ algorithm is meaningless for hiding information of $\mathsf{der}\text{-}kw$. More concretely, from the challenge trapdoor $\mathsf{td}^*_{\mathsf{S}[i^*],kw_b^*} = \mathsf{td}_{\mathsf{der}\text{-}kw_b^*}$ where $\mathsf{der}\text{-}kw_b^*$ is the extended keyword for kw_b^*, information of $\mathsf{der}\text{-}kw_b^*$ is leaked. Thus, for providing IND-IKGA security, we need to guarantee that information of kw is not leaked from the corresponding extended keyword $\mathsf{der}\text{-}kw$. Fortunately, we have already showed that $\mathsf{der}\text{-}kw_b^*$ is indistinguishable from random. More concretely, in Game_5 of the proof of Theorem 2, the distribution of $\mathsf{der}\text{-}kw_b^*$ is identical when $b = 0$ and $b = 1$, respectively. This is sufficient to provide IND-IKGA security and we omit the proof. We remark that adaptive smoothness does not directly guarantee unlinkability that hides information whether two hash values are computed by the same input or not. Thus, the adversary $\mathcal{A}$ may distinguish $b = 0$ or $b = 1$ if $\mathcal{A}$ obtains a trapdoor for (kw_0^*, i^*) (or (kw_1^*, i^*)), and obtains (information of) $\mathsf{der}\text{-}kw_0^*$ (or $\mathsf{der}\text{-}kw_1^*$). Thus, we have restricted that $\mathcal{A}$ is allowed to issue a trapdoor query $(kw, i) \notin \{(kw_0^*, i^*), (kw_1^*, i^*)\}$ in the security definition.

6 Conclusion and Future Work

In this paper, we proposed a generic construction of PAEKS from $\mathsf{WI\text{-}SPHF}$, PKE, PRF, and PEKS. We employ the Katz-Vaikuntanathan methodology that allows two entities can share a key non-interactively by employing word-independent SPHF. Our construction is not only more efficient than the Liu et al. construction but also provides stronger security than that of Liu et al.

As mentioned in the introduction, PAEKS is one option for providing trapdoor privacy, and designated-tester PEKS is another option. As in Abdalla et al. generic construction of PEKS [1], designated-tester PEKS can be generically constructed from anonymous IBE [16, 31]. While some schemes claimed that they provide trapdoor privacy, these generic constructions do not provide trapdoor privacy owing to their construction methodology. That is, as in the Abdalla et al. generic construction of PEKS, a trapdoor is a secret key of anonymous IBE (a keyword is regarded as identity). Since the secret key generation of IBE can be regarded as a signing algorithm (Naor transformation [8]), a trapdoor leaks information of keyword.[8] Noroozi and Eslami [27] also proposed a generic transformation of designated-tester PEKS which is claimed to be secure against *online* keyword guessing attacks, from re-randomizable PKE and designated-tester PEKS secure against *offline* keyword guessing attacks. Here, the offline keyword guessing attack is the attack that we explained in the introduction.[9] That is, they assumed that the underlying designated-tester PEKS provides trapdoor privacy. Thus, still no generic construction of designated-tester PEKS with trapdoor privacy has been proposed so far. Thus, considering whether the Katz-Vaikuntanathan methodology can be employed for constructing trapdoor private designated-tester PEKS or not is an interesting future work.

Acknowledgment: The author would like to thank anonymous reviewers of ACM APKC 2022 for their invaluable comments and suggestions. This work was supported by JSPS KAKENHI Grant Number JP21K11897.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *Journal of Cryptology*, 21(3):350–391, 2008.

[2] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[3] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *ICCSA*, pages 1249–1259, 2008.

[4] Rouzbeh Behnia, Muslum Ozgur Ozmen, and Attila Altay Yavuz. Lattice-based public key searchable encryption from experimental perspectives. *IEEE Transactions on Dependable and Secure Computing*, 17(6):1269–1282, 2020.

[8] Chen [10] modified the syntax where a trapdoor generation algorithm takes a server public key as input. Then, a receiver encrypts a trapdoor by using the server public key and sends the ciphertext to the server (i.e., trapdoors are ciphertexts). The server still can run the test algorithm by decrypting the ciphertext and obtaining the trapdoor. This simple modification works well since no information of keyword is leaked owing to the encryption. Here, the server is not considered as an adversary. This method essentially establishes a secure channel between the server and the receiver.

[9] The online keyword guessing attack is explained as follows. An adversary chooses a keyword kw, generates its ciphertext, say CT, and sends CT to the server. When the receiver sends a trapdoor to the server, the adversary can check whether the trapdoor is associated to kw or not by checking whether the server returns CT to the receiver. This attack can be easily prevented if ciphertexts are re-randomizable, i.e., the server re-randomizes all ciphertexts before answering the search query.

[5] Fabrice Benhamouda, Olivier Blazy, Céline Chevalier, David Pointcheval, and Damien Vergnaud. New techniques for SPHFs and efficient one-round PAKE protocols. In *CRYPTO*, pages 449–475, 2013.

[6] Fabrice Benhamouda, Olivier Blazy, Léo Ducas, and Willy Quach. Hash proof systems over lattices revisited. In *Public-Key Cryptography*, pages 644–674, 2018.

[7] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[9] Dan Boneh, Ananth Raghunathan, and Gil Segev. Function-private identity-based encryption: Hiding the function in functional encryption. In *CRYPTO*, pages 461–478, 2013.

[10] Yu-Chi Chen. SPEKS: secure server-designation public key encryption with keyword search against keyword guessing attacks. *The Computer Journal*, 58(4):922–933, 2015.

[11] Leixiao Cheng and Fei Meng. Security analysis of Pan et al.'s "public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability". *Journal of Systems Architecture*, 119:102248, 2021.

[12] Tianyu Chi, Baodong Qin, and Dong Zheng. An efficient searchable public-key authenticated encryption for cloud-assisted medical internet of things. *Wireless Communications and Mobile Computing*, 2020:8816172:1–8816172:11, 2020.

[13] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *EUROCRYPT*, pages 45–64, 2002.

[14] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Review*, 45(4):727–784, 2003.

[15] Léo Ducas, Vadim Lyubashevsky, and Thomas Prest. Efficient identity-based encryption over NTRU lattices. In *ASIACRYPT*, pages 22–41, 2014.

[16] Keita Emura, Atsuko Miyaji, Mohammad Shahriar Rahman, and Kazumasa Omote. Generic constructions of secure-channel free searchable encryption with adaptive security. *Security and Communication Networks*, 8(8):1547–1560, 2015.

[17] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. A secure channel free public key encryption with keyword search scheme without random oracle. In *CANS*, pages 248–258, 2009.

[18] Liming Fang, Willy Susilo, Chunpeng Ge, and Jiandong Wang. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Information Sciences*, 238:221–241, 2013.

[19] Chunxiang Gu, Yuefei Zhu, and Heng Pan. Efficient public key encryption with keyword search schemes from pairings. In *Inscrypt*, pages 372–383, 2007.

[20] Qiong Huang and Hongbo Li. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences*, 403:1–14, 2017.

[21] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *TCC*, pages 293–310, 2011.

[22] Zengpeng Li and Ding Wang. Achieving one-round password-based authenticated key exchange over lattices. *IEEE Transactions on Services Computing*, 2019. Early Access.

[23] Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. Public-Key Authenticated Encryption with Keyword Search: A Generic Construction and Its Quantum-Resistant Instantiation. *The Computer Journal*, 09 2021.

[24] Zi-Yuan Liu, Yi-Fan Tseng, Raylin Tso, Masahiro Mambo, and Yu-Chi Chen. Public-key authenticated encryption with keyword search: Cryptanalysis, enhanced security, and quantum-resistant instantiation. *IACR Cryptology ePrint Archive*, page 1008, 2021. Version 3, posted on 23-Nov-2021. ACM ASIACCS 2022, to appear.

[25] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.

[26] Mahnaz Noroozi and Ziba Eslami. Public key authenticated encryption with keyword search: revisited. *IET Information Security*, 13(4):336–342, 2019.

[27] Mahnaz Noroozi and Ziba Eslami. Public-key encryption with keyword search: a generic construction secure against online and offline keyword guessing attacks. *Journal of Ambient Intelligence and Humanized Computing*, 11(2):879–890, 2020.

[28] Xiangyu Pan and Fagen Li. Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. *Journal of Systems Architecture*, 115:102075, 2021.

[29] Baodong Qin, Yu Chen, Qiong Huang, Ximeng Liu, and Dong Zheng. Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Information Sciences*, 516:515–528, 2020.

[30] Baodong Qin, Hui Cui, Xiaokun Zheng, and Dong Zheng. Improved security model for public-key authenticated encryption with keyword search. In *ProvSec*, pages 19–38, 2021.

[31] Hyun Sook Rhee, Jong Hwan Park, and Dong Hoon Lee. Generic construction of designated tester public-key encryption with keyword search. *Information Sciences*, 205:93–109, 2012.

[32] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Improved searchable public key encryption with designated tester. In *ACM ASIACCS*, pages 376–379, 2009.

[33] Hyun Sook Rhee, Willy Susilo, and Hyun-Jeong Kim. Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electron. Express*, 6(5):237–243, 2009.