Generic Construction of Dual-Server Public Key Authenticated Encryption with Keyword Search

Keita Emura[§]

[§]Kanazawa University, Japan.[*]

June 19, 2024

Abstract

Chen et al. (IEEE Transactions on Cloud Computing 2022) introduced dual-server public key authenticated encryption with keyword search (DS-PAEKS), and proposed a DS-PAEKS scheme under the decisional Diffie-Hellman assumption. In this paper, we propose a generic construction of DS-PAEKS from PAEKS, public key encryption, and signatures. By providing a concrete attack, we show that the DS-PAEKS scheme of Chen et al. is vulnerable. That is, the proposed generic construction yields the first DS-PAEKS schemes. Our attack with a slight modification works against the Chen et al. dual-server public key encryption with keyword search (DS-PEKS) scheme (IEEE Transactions on Information Forensics and Security 2016). Moreover, we demonstrate that the Tso et al. generic construction of DS-PEKS from public key encryption (IEEE Access 2020) is also vulnerable. We also analyze other pairing-free PAEKS schemes (Du et al., Wireless Communications and Mobile Computing 2022 and Lu and Li, IEEE Transactions on Mobile Computing 2022). Though we did not find any attack against these schemes, we show that at least their security proofs are wrong.

1 Introduction

Public key encryption with keyword search (PEKS) [1] provides a search functionality over encrypted data in a public key setting. A sender encrypts a keyword kw using the public key of a receiver. The receiver then generates a trapdoor for a keyword kw' using the secret key of the receiver. The test algorithm that takes a ciphertext and a trapdoor as input outputs 1 if $kw = kw'$. Similar to correctness, (computational) consistency is defined, where no probabilistic polynomial-time (PPT) adversary can produce kw and kw' such that $kw \neq kw'$ and the test algorithm outputs 1 with a ciphertext of kw and a trapdoor of kw'. It is required that no information about keywords is revealed from ciphertexts. However, information about which keyword is associated with the trapdoor is leaked by running a test algorithm with self-made ciphertexts. Anyone can generate a ciphertext; hence, the keyword guessing attack is unavoidable in PEKS. To prevent the keyword guessing attack, public key authenticated encryption with keyword search (PAEKS) [3,7,8,10,12,13,15,17–21,23] has been proposed, where a sender secret key is required for encryption. PAEKS requires that no information about the keyword is leaked from both ciphertexts and trapdoors.

[*]The main part of study was done when the author was with the National Institute of Information and Communications Technology (NICT), Japan.

Chen et al. [4] further extended PAEKS by introducing a dual-server setting,[1] which they call dual-server PAEKS (DS-PAEKS). In DS-PAEKS, there are two servers, the assistant server and the test server that manage their own public and secret keys, respectively. DS-PAEKS can be regarded as an extension of dual-server PEKS (DS-PEKS) [5] which does not require the secret key of the sender for encryption. The DS-PAEKS flow is described below. A sender encrypts a keyword kw using the secret key of the sender $\mathsf{sk_S}$ and the public keys of a receiver $\mathsf{pk_R}$, assistant server $\mathsf{pk_{AS}}$, and test server $\mathsf{pk_{TS}}$ and uploads the ciphertext $\mathsf{ct_{DS\text{-}PAEKS}}$ to the assistant server. A receiver generates a trapdoor $\mathsf{td}_{kw'}$ for a keyword kw' using the secret key of the receiver $\mathsf{sk_R}$ and the public keys of a sender $\mathsf{pk_S}$, assistant server $\mathsf{pk_{AS}}$, and test server $\mathsf{pk_{TS}}$, and uploads $\mathsf{td}_{kw'}$ to the assistant server. The assistant server converts the ciphertext and the trapdoor to an intermediate ciphertext $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ via the transition algorithm using the secret key of the assistant server $\mathsf{sk_{AS}}$, and sends $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ to the test server. Finally, the test server runs the test algorithm, that takes the intermediate ciphertext $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ and the secret key of the test server $\mathsf{sk_{TS}}$ as input. Chen et al. claimed that the dual-server setting prevents running the test algorithm by a single server that prevents a keyword guessing attack. That is, in PAEKS, the cases that an adversary trivially wins are excluded in the security definitions, and thus, if a server that runs the test algorithm obtains a ciphertext and a trapdoor of the challenge keyword for the same sender, then there is no way to prevent keyword guessing attacks in PAEKS. By introducing dual servers, there is room for protecting keyword guessing attacks in more strict way. For example, for an adversary that is modeled as a malicious assistant server, it is guaranteed that no information about the keyword is leaked from the challenge ciphertext, even if the adversary obtains a trapdoor for the challenge keyword, and converts the challenge ciphertext and the trapdoor. Similarly, for an adversary that is modeled as a malicious test server, it is guaranteed that no information about the keyword is leaked from the challenge ciphertext, even if the adversary obtains the corresponding intermediate ciphertext converted from the challenge ciphertext and trapdoor. Here, it is assumed that two servers do not collude.

Chen et al. gave a formal security definition of DS-PAEKS and proposed the DS-PAEKS scheme under the decisional Diffie-Hellman (DDH) assumption. However, the following restrictions in their security definitions can be observed:

- An adversary that is modeled as a malicious assistant server is allowed to issue any query, including challenge keywords, to the encryption, trapdoor, and test oracles.
  - Constructing a DS-PAEKS scheme, which is secure in this definition, is impossible because of the following general attack: An adversary that has the secret key of the assistant server $\mathsf{sk_{AS}}$ issues a challenge keyword kw_0^* to the trapdoor oracle. After obtaining the challenge ciphertext $\mathsf{ct^*_{DS\text{-}PAEKS}}$, the adversary prepares an intermediate ciphertext $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ from $\mathsf{ct^*_{DS\text{-}PAEKS}}$, trapdoor $\mathsf{td}_{kw_0^*}$, and $\mathsf{sk_{AS}}$, and sends $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ to the test oracle. If $\mathsf{ct^*_{DS\text{-}PAEKS}}$ is an encryption of kw_0^*, then the test oracle returns 1, and 0 if $\mathsf{ct^*_{DS\text{-}PAEKS}}$ is an encryption of kw_1^*. This completely breaks the security.
  - Even if the adversary is not allowed to query the challenge keywords to the trapdoor oracle, the DS-PAEKS scheme of Chen et al. is vulnerable. Briefly, the adversary can prepare an intermediate ciphertext of the challenge keyword from $\mathsf{sk_{AS}}$, $\mathsf{ct^*_{DS\text{-}PAEKS}}$, and a ciphertext of the challenge keyword obtained via the encryption oracle. We demonstrate

[1]Cheng and Meng [9] proposed server-aided PAEKS (SA-PAEKS). Though it also introduces two servers, the roles of these servers are different from those of DS-PAEKS. In SA-PAEKS, these servers are called a sender server and a receiver server, and they are related to encryption and trapdoor generation, whereas servers are related to searching in DS-PAEKS.

the attack in Section 4. Our attack with a slight modification also works against the Chen et al. DS-PEKS scheme [5].

Our Contribution. In this paper, we propose a generic construction of DS-PAEKS derived from PAEKS, two PKE schemes, and two signature schemes. We also introduce a new security definition of DS-PAEKS that considers the general attack above. As concrete instantiations of the proposed generic construction, we can employ the Qin et al. pairing-based PAEKS scheme [20] or the Cheng-Meng lattice-based PAEKS scheme [8] with appropriate PKE and signature schemes.

We also give a concrete attack against the Chen et al. DS-PAEKS scheme [4]. We emphasize that the attack is done according to their security model where an adversary obtains the assistant server's secret key $\mathsf{sk}_{\mathsf{AS}}$ and is allowed to access the test oracle. That is, the proposed generic construction yields the first DS-PAEKS schemes. Our attack with a slight modification works against the Chen et al. DS-PEKS scheme [5]. Moreover, we demonstrate that a generic construction of DS-PEKS from PKE [22] is vulnerable. We also analyze other pairing-free PAEKS schemes [11, 16]. Though we did not find any attack against these schemes, we show that at least their security proofs are wrong.

2 Preliminaries

2.1 PKE and Signature

PKE. Let $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme. The key generation algorithm $\mathsf{PKE.KeyGen}$ takes a security parameter λ as input, and outputs a key pair $(\mathsf{PK}, \mathsf{DK})$. The encryption algorithm $\mathsf{PKE.Enc}$ takes PK and a plaintext M, and outputs a ciphertext C. The decryption algorithm $\mathsf{PKE.Dec}$ takes DK and C, and outputs M or $\perp$. We require that PKE provides indistinguishability against the chosen-ciphertext attack (IND-CCA), where an PPT adversary $\mathcal{A}$ is allowed to issue decryption queries $C \neq C^*$ where C^* is the challenge ciphertext that is an encryption of either M_0^* or M_1^*. $\mathcal{A}$ wins if $\mathcal{A}$ can distinguish whether C^* is an encryption of M_0^* or M_1^*.

Signature. Let $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. The key generation algorithm $\mathsf{Sig.KeyGen}$ takes a security parameter λ, and outputs a key pair $(\mathsf{vk}, \mathsf{sigk})$. The signing algorithm Sign takes sigk and a message M as input, and outputs a signature σ. Here, we explicitly assume that the Sign algorithm is probabilistic (See the proof of Lemma 3). The verification algorithm Verify takes vk, σ, and M as input, and outputs 0 or 1. We require that Sig provides strongly existential unforgeability under the adaptive chosen message attack (sEUF-CMA), where a PPT adversary $\mathcal{A}$ is allowed to issue a signing query M and obtains $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}, M)$. (M, σ) is then preserved to a set Set. $\mathcal{A}$ wins if $\mathcal{A}$ can produce (M^*, σ^*), where $\mathsf{Verify}(\mathsf{vk}, \sigma^*, M^*) = 1$ and $(M^*, \sigma^*) \notin \mathsf{Set}$.

2.2 PAEKS

Definition 1 (Syntax of PAEKS). *A PAEKS scheme* PAEKS *consists of the six algorithms* $(\mathsf{PAEKS.Setup}, \mathsf{PAEKS.KG_R}, \mathsf{PAEKS.KG_S}, \mathsf{PAEKS.Enc}, \mathsf{PAEKS.Trapdoor}, \mathsf{PAEKS.Test})$ *defined as follows.*

$\mathsf{PAEKS.Setup}$: *The setup algorithm takes a security parameter* λ *as input, and outputs a common parameter* pp*. We assume that* pp *implicitly contains the keyword space* $\mathcal{KS}$*.*

$\mathsf{PAEKS.KG_R}$**:** *The receiver key generation algorithm takes* pp *as input, and outputs a public key* $\mathsf{pk_R}$ *and secret key* $\mathsf{sk_R}$*.*

$\mathsf{PAEKS.KG_S}$**:** *The sender key generation algorithm takes* pp *as input, and outputs a public key* $\mathsf{pk_S}$ *and secret key* $\mathsf{sk_S}$*.*

$\mathsf{PAEKS.Enc}$**:** *The keyword encryption algorithm takes* $\mathsf{pk_R}$*,* $\mathsf{pk_S}$*,* $\mathsf{sk_S}$*, and a keyword* $kw \in \mathcal{KS}$ *as input, and outputs a ciphertext* $\mathsf{ct_{PAEKS}}$*.*

$\mathsf{PAEKS.Trapdoor}$**:** *The trapdoor algorithm takes* $\mathsf{pk_R}$*,* $\mathsf{pk_S}$*,* $\mathsf{sk_R}$*, and a keyword* $kw' \in \mathcal{KS}$ *as input, and outputs a trapdoor* $\mathsf{td}_{kw'}$*.*

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

Definition 2 (Correctness)**.** *For any security parameter* λ*, any common parameter* $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$*, any key pair* $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp})$ *and* $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})$*, and any keyword* $kw \in \mathcal{KS}$*, let* $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, kw)$ *and* $\mathsf{td}_{kw} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw)$*. Then* $\Pr[\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}_{kw}) = 1] = 1 - \mathsf{negl}(\lambda)$ *holds.*

Definition 3 (Computational Consistency)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})\\
&\quad (kw, kw') \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S})\ \textit{s.t.}\ kw, kw' \in \mathcal{KS} \wedge kw \neq kw'\\
&\quad \mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, kw)\\
&\quad \mathsf{td}_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw')\\
&\quad \textit{If}\ \mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}_{kw'}) = 1,\ \textit{then output}\ 1,\ \textit{and}\ 0\ \textit{otherwise.}
\end{aligned}
$$

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 *for all PPT adversaries* $\mathcal{A}$*.*

Next, we define the indistinguishability against the chosen keyword attack (IND-CKA) and that against the inside keyword guessing attack (IND-IKGA), which ensure that no information about the keyword is leaked from ciphertexts or trapdoors, respectively.

Definition 4 (IND-CKA)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})\\
&\quad (kw_0^*, kw_1^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S})\ \textit{s.t.}\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^*\\
&\quad \beta \xleftarrow{\$} \{0,1\};\ \mathsf{ct}^*_{\mathsf{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, kw_\beta^*)\\
&\quad \beta' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{ct}^*_{\mathsf{PAEKS}})\\
&\quad \textit{If}\ \beta = \beta'\ \textit{then output}\ 1,\ \textit{and}\ 0\ \textit{otherwise.}
\end{aligned}
$$

Here, $\mathcal{O} := \{\mathcal{O}_C(\cdot), \mathcal{O}_{Trap}(\cdot)\}$*.* $\mathcal{O}_C$ *takes* $kw \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, kw)$*. Here, there is no restriction.* $\mathcal{O}_{Trap}$ *takes* $kw' \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw')$*. Here,* $kw' \notin \{kw_0^*, kw_1^*\}$*.* PAEKS *is IND-CKA secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1] - 1/2|$ *is negligible in the security parameter* λ *for all PPT adversaries* $\mathcal{A}$*.*

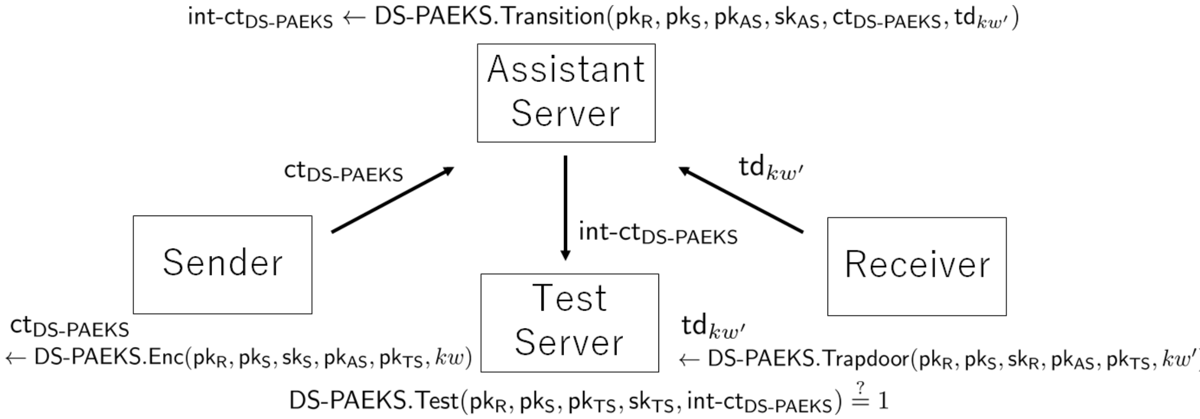

Figure 1: DS-PAEKS

Definition 5 (IND-IKGA)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}IKGA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})\\
&\quad (kw_0^*, kw_1^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S})\ \ s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^*\\
&\quad \beta \stackrel{\$}{\leftarrow} \{0,1\};\ \mathsf{td}^*_{kw^*_\beta} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw^*_\beta)\\
&\qquad\qquad \beta' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{td}^*_{kw^*_\beta})\\
&\qquad\qquad \textit{If } \beta = \beta' \textit{ then output } 1,\ \textit{and } 0 \textit{ otherwise.}
\end{aligned}
$$

Here, $\mathcal{O} := \{\mathcal{O}_C(\cdot), \mathcal{O}_{Trap}(\cdot)\}$*.* $\mathcal{O}_C$ *takes* $kw \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, kw)$*. Here,* $kw \notin \{kw_0^*, kw_1^*\}$*.* $\mathcal{O}_{Trap}$ *takes* $kw' \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, kw')$*. Here* $kw' \notin \{kw_0^*, 0, kw_1^*, 0\}$*.* PAEKS *is IND-IKGA secure if the advantage* $\mathsf{Adv}^{\mathsf{IND\text{-}IKGA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}IKGA}}_{\mathsf{PAEKS},\mathcal{A}}(\lambda) = 1] - 1/2|$ *is negligible in the security parameter* λ *for all PPT adversaries* $\mathcal{A}$*.*

3 Definitions of DS-PAEKS

In this section, we introduce the DS-PAEKS definitions. As mentioned in the Introduction, the definitions given in [14] were not well defined because there is a general attack. Thus, we newly introduce the DS-PAEKS definitions. Figure 1 describes the DS-PAEKS flow.

Definition 6 (Syntax of DS-PAEKS)**.** *A DS-PAEKS scheme* DS-PAEKS *consists of the nine algorithms* ($\mathsf{DS\text{-}PAEKS.Setup}$, $\mathsf{DS\text{-}PAEKS.KG_R}$, $\mathsf{DS\text{-}PAEKS.KG_S}$, $\mathsf{DS\text{-}PAEKS.KG_{AS}}$, $\mathsf{DS\text{-}PAEKS.KG_{TS}}$, DS-PAEKS.Enc, DS-PAEKS.Trapdoor, DS-PAEKS.Transition, DS-PAEKS.Test) *defined as follows.*

DS-PAEKS.Setup: *The setup algorithm takes a security parameter* λ *as input, and outputs a common parameter* pp*. We assume that* pp *implicitly contains the keyword space* $\mathcal{KS}$*.*

$\mathsf{DS\text{-}PAEKS.KG_R}$: *The receiver key generation algorithm takes* pp *as input, and outputs a public key* $\mathsf{pk_R}$ *and secret key* $\mathsf{sk_R}$*.*

$\mathsf{DS\text{-}PAEKS.KG_S}$**:** *The sender key generation algorithm takes* pp *as input, and outputs a public key* $\mathsf{pk_S}$ *and secret key* $\mathsf{sk_S}$*.*

$\mathsf{DS\text{-}PAEKS.KG_{AS}}$**:** *The assistant server key generation algorithm takes* pp *as input, and outputs a public key* $\mathsf{pk_{AS}}$ *and secret key* $\mathsf{sk_{AS}}$*.*

$\mathsf{DS\text{-}PAEKS.KG_{TS}}$**:** *The test server key generation algorithm takes* pp *as input, and outputs a public key* $\mathsf{pk_{TS}}$ *and secret key* $\mathsf{sk_{TS}}$*.*

$\mathsf{DS\text{-}PAEKS.Enc}$**:** *The keyword encryption algorithm takes* $\mathsf{pk_R}$, $\mathsf{pk_S}$, $\mathsf{sk_S}$, $\mathsf{pk_{AS}}$, $\mathsf{pk_{TS}}$, *and a keyword* $kw \in \mathcal{KS}$ *as input, and outputs a ciphertext* $\mathsf{ct_{DS\text{-}PAEKS}}$*.*

$\mathsf{DS\text{-}PAEKS.Trapdoor}$**:** *The trapdoor algorithm takes* $\mathsf{pk_R}$, $\mathsf{pk_S}$, $\mathsf{sk_R}$, $\mathsf{pk_{AS}}$, $\mathsf{pk_{TS}}$, *and a keyword* $kw' \in \mathcal{KS}$ *as input, and outputs a trapdoor* $\mathsf{td}_{kw'}$*.*

$\mathsf{DS\text{-}PAEKS.Transition}$**:** *The transition algorithm takes* $\mathsf{pk_R}$, $\mathsf{pk_S}$, $\mathsf{pk_{AS}}$, $\mathsf{sk_{AS}}$, $\mathsf{ct_{DS\text{-}PAEKS}}$, *and* $\mathsf{td}_{kw'}$ *as input, and outputs an intermediate ciphertext* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$*.*

$\mathsf{DS\text{-}PAEKS.Test}$**:** *The test algorithm takes* $\mathsf{pk_R}$, $\mathsf{pk_S}$, $\mathsf{pk_{TS}}$, $\mathsf{sk_{TS}}$, *and* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ *as input, and outputs 1 or 0.*

Definition 7 (Correctness)**.** *For any security parameter* λ*, any common parameter* $\mathsf{pp} \leftarrow \mathsf{DS\text{-}PAEKS.Setup}(1^\lambda)$*, any key pair* $(\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp})$*,* $(\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})$*,* $(\mathsf{pk_{AS}}, \mathsf{sk_{AS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp})$*, and* $(\mathsf{pk_{TS}}, \mathsf{sk_{TS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})$*, and any keyword* $kw \in \mathcal{KS}$*, let* $\mathsf{ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$ *and* $\mathsf{td}_{kw} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$*. Then, for* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct_{DS\text{-}PAEKS}}, \mathsf{td}_{kw})$*,* $\Pr[\mathsf{DS\text{-}PAEKS.Test}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}}, \mathsf{int\text{-}ct_{DS\text{-}PAEKS}}) = 1] = 1 - \mathsf{negl}(\lambda)$ *holds.*

Definition 8 (Computational Consistency)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{DS\text{-}PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})\\
&\quad (\mathsf{pk_{AS}}, \mathsf{sk_{AS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp});\ (\mathsf{pk_{TS}}, \mathsf{sk_{TS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})\\
&\quad (kw, kw') \leftarrow \mathcal{A}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}})\ \textit{s.t. } kw, kw' \in \mathcal{KS} \land kw \neq kw'\\
&\quad \mathsf{ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)\\
&\quad \mathsf{td}_{kw'} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw')\\
&\quad \mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct_{DS\text{-}PAEKS}}, \mathsf{td}_{kw'})\\
&\quad \textit{If } \mathsf{DS\text{-}PAEKS.Test}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}}, \mathsf{int\text{-}ct_{DS\text{-}PAEKS}}) = 1,\\
&\quad\ \textit{then output } 1,\ \textit{and } 0\ \textit{otherwise.}
\end{aligned}
$$

$\mathsf{DS\text{-}PAEKS}$ *is consistent if the advantage*

$$\mathsf{Adv}^{\mathsf{consist}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathsf{consist}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = 1]$$

is negligible in the security parameter λ *for all PPT adversaries* $\mathcal{A}$*.*

Next, we define IND-CKA for the assistant server (IND-AS-CKA), where the adversary is given $\mathsf{sk_{AS}}$. Considering the role of the assistant server, we must guarantee that no information about the keyword is leaked from the challenge ciphertext, even if the adversary obtains a trapdoor for the challenge keyword, and runs the $\mathsf{DS\text{-}PAEKS.Transition}$ algorithm with the challenge ciphertext and the trapdoor. However, if there is no restriction, then the adversary can trivially break the IND-AS-CKA security, i.e., by using $\mathsf{sk_{AS}}$, the adversary generates an intermediate ciphertext from the challenge ciphertext and a trapdoor of either kw_0^* or kw_1^*, and sends the intermediate ciphertext to the test oracle $\mathcal{O}_{\mathrm{Test}}$. Thus, we introduce the following restriction: the adversary is allowed to issue $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ to $\mathcal{O}_{\mathrm{Test}}$ where $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \notin \{\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \mid \mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct^*_{DS\text{-}PAEKS}}, \mathsf{td}_{kw}) \wedge \mathsf{td}_{kw} \in \mathsf{TSet}\}$. Here, TSet is a set of trapdoors for the challenge keywords kw_0^* and kw_1^*. We remark that kw_0^* and kw_1^* are declared during the challenge phase. Thus, TSet is defined after the challenge phase.

Definition 9 (IND-AS-CKA)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}AS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{DS\text{-}PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})\\
&\quad (\mathsf{pk_{AS}}, \mathsf{sk_{AS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp});\ (\mathsf{pk_{TS}}, \mathsf{sk_{TS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})\\
&\quad \mathsf{TSet} := \emptyset\\
&\quad (kw_0^*, kw_1^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{pk_{TS}})\\
&\qquad s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^*\\
&\quad \beta \xleftarrow{\$} \{0,1\};\ \mathsf{ct^*_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw_\beta^*)\\
&\quad \beta' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{ct^*_{DS\text{-}PAEKS}})\\
&\quad \textit{If } \beta = \beta' \textit{ then output } 1,\ \textit{and } 0 \textit{ otherwise.}
\end{aligned}
$$

Here, $\mathcal{O} := \{\mathcal{O}_C(\cdot), \mathcal{O}_{Trap}(\cdot), \mathcal{O}_{Test}(\cdot)\}$*.* $\mathcal{O}_C$ *takes* $kw \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$*. Here, there is no restriction.* $\mathcal{O}_{Trap}$ *takes* $kw' \in \mathcal{KS}$ *as input, and returns* $\mathsf{td}_{kw'} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw')$*. Here, there is no restriction, i.e., the challenge keywords can be queried. If* $kw \in \{kw_0^*, kw_1^*\}$*, then* $\mathsf{TSet} := \mathsf{TSet} \cup \{\mathsf{td}_{kw'}\}$*. We note that in the challenge phase,* TSet *is updated by the trapdoors of* $kw \in \{kw_0^*, kw_1^*\}$ *which are generated before* $\mathcal{A}$ *declares* (kw_0^*, kw_1^*)*.* $\mathcal{O}_{Test}$ *takes* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$*, and returns the result of* $\mathsf{DS\text{-}PAEKS.Test}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}}, \mathsf{int\text{-}ct_{DS\text{-}PAEKS}})$*. Here, we restrict that* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \notin \{\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \mid \mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct^*_{DS\text{-}PAEKS}}, \mathsf{td}_{kw}) \wedge \mathsf{td}_{kw} \in \mathsf{TSet}\}$*.* $\mathsf{DS\text{-}PAEKS}$ *is IND-AS-CKA secure if the advantage*

$$\mathsf{Adv}^{\mathsf{IND\text{-}AS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}AS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in the security parameter λ *for all PPT adversaries* $\mathcal{A}$*.*

Next, we define IND-CKA for the test server (IND-TS-CKA), where the adversary is given $\mathsf{sk_{TS}}$. Considering the role of the test server, we must guarantee that no information about the keyword is leaked from the challenge ciphertext, even if the corresponding intermediate ciphertext is given. However, if the adversary is allowed to obtain a trapdoor for the challenge keyword, the adversary can trivially break the IND-TS-CKA security. Thus, we restrict the input of the trapdoor oracle $\mathcal{O}_{\mathrm{Trap}}$ as $kw' \notin \{kw_0^*, kw_1^*\}$.

Definition 10 (IND-TS-CKA)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}TS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{DS\text{-}PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})\\
&\quad (\mathsf{pk_{AS}}, \mathsf{sk_{AS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp});\ (\mathsf{pk_{TS}}, \mathsf{sk_{TS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})\\
&\quad (kw_0^*, kw_1^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}})\\
&\quad\ \ s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^*\\
&\quad \beta \xleftarrow{\$} \{0,1\};\ \mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw^*_\beta)\\
&\quad \beta' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}})\\
&\quad \textit{If } \beta = \beta' \textit{ then output } 1,\ \textit{and } 0 \textit{ otherwise.}
\end{aligned}
$$

Here, $\mathcal{O} := \{\mathcal{O}_C(\cdot), \mathcal{O}_{Trap}(\cdot), \mathcal{O}_{Trans}(\cdot,\cdot)\}$*.* $\mathcal{O}_C$ *takes* $kw \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$*. Here, there is no restriction.* $\mathcal{O}_{Trap}$ *takes* $kw' \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw')$*. Here,* $kw' \notin \{kw_0^*, kw_1^*\}$*.* $\mathcal{O}_{Trans}$ *takes* $\mathsf{ct_{DS\text{-}PAEKS}}$ *and* td_{kw}*, and returns the result of* $\mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct_{DS\text{-}PAEKS}}, \mathsf{td}_{kw})$*. Here, there is no restriction.* DS-PAEKS *is IND-TS-CKA secure if the advantage*

$$\mathsf{Adv}^{\mathsf{IND\text{-}TS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}TS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in the security parameter λ *for all PPT adversaries* $\mathcal{A}$*.*

Next, we define IND-IKGA for the assistant server (IND-AS-IKGA) where the adversary is given $\mathsf{sk_{AS}}$.

Definition 11 (IND-AS-IKGA)**.** *We define the experiment:*

$$
\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}AS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{DS\text{-}PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})\\
&\quad (\mathsf{pk_{AS}}, \mathsf{sk_{AS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp});\ (\mathsf{pk_{TS}}, \mathsf{sk_{TS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})\\
&\quad \mathsf{CTSet} := \emptyset\\
&\quad (kw_0^*, kw_1^*, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{pk_{TS}})\\
&\quad\ \ s.t.\ kw_0^*, kw_1^* \in \mathcal{KS} \wedge\ kw_0^* \neq kw_1^*\\
&\quad \beta \xleftarrow{\$} \{0,1\};\ \mathsf{td}^*_{kw^*_\beta} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw^*_\beta)\\
&\quad \beta' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{td}^*_{kw^*_\beta})\\
&\quad \textit{If } \beta = \beta' \textit{ then output } 1,\ \textit{and } 0 \textit{ otherwise.}
\end{aligned}
$$

Here, $\mathcal{O} := \{\mathcal{O}_C(\cdot), \mathcal{O}_{Trap}(\cdot), \mathcal{O}_{Test}(\cdot)\}$*.* $\mathcal{O}_C$ *takes* $kw \in \mathcal{KS}$ *as input, and returns* $\mathsf{ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$*. Here, there is no restriction. If* $kw \in \{kw_0^*, kw_1^*\}$*, then* $\mathsf{CTSet} := \mathsf{CTSet} \cup \{\mathsf{ct_{DS\text{-}PAEKS}}\}$*. We note that in the challenge phase,* CTSet *is updated by the ciphertexts of* $kw \in \{kw_0^*, kw_1^*\}$ *generated before* $\mathcal{A}$ *declares* (kw_0^*, kw_1^*)*.* $\mathcal{O}_{Trap}$ *takes* $kw' \in \mathcal{KS}$ *as input, and returns the result of* $\mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw')$*. Here,* $kw' \notin \{kw_0^*, kw_1^*\}$*.* $\mathcal{O}_{Test}$ *takes* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$*, and returns the result of* $\mathsf{DS\text{-}PAEKS.Test}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}}, \mathsf{int\text{-}ct_{DS\text{-}PAEKS}})$*. Here,* $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \notin \{\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \mid \mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R},$

$\mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct_{DS\text{-}PAEKS}}, \mathsf{td}^*_{kw^*_\beta}) \wedge \mathsf{ct_{DS\text{-}PAEKS}} \in \mathsf{CTSet}\}$. DS-PAEKS *is IND-AS-IKGA secure if the advantage*

$$\mathsf{Adv}^{\mathsf{IND\text{-}AS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}AS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$.

Finally, we define IND-IKGA security for the test server (IND-TS-IKGA), where the adversary is given $\mathsf{sk_{TS}}$.

Definition 12 (IND-TS-IKGA)**.** *We define the experiment:*

$$\begin{aligned}
&\mathsf{Exp}^{\mathsf{IND\text{-}TS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda):\\
&\quad \mathsf{pp} \leftarrow \mathsf{DS\text{-}PAEKS.Setup}(1^\lambda)\\
&\quad (\mathsf{pk_R}, \mathsf{sk_R}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp});\ (\mathsf{pk_S}, \mathsf{sk_S}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})\\
&\quad (\mathsf{pk_{AS}}, \mathsf{sk_{AS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp});\ (\mathsf{pk_{TS}}, \mathsf{sk_{TS}}) \leftarrow \mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})\\
&\quad (kw^*_0, kw^*_1, \mathsf{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{pk_{TS}})\\
&\quad\ \ s.t.\ kw^*_0, kw^*_1 \in \mathcal{KS} \wedge\ kw^*_0 \neq kw^*_1\\
&\quad \beta \xleftarrow{\$} \{0,1\};\ \mathsf{td}^*_{kw^*_\beta} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw^*_\beta)\\
&\quad \beta' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{state}, \mathsf{td}^*_{kw^*_\beta})\\
&\quad \textit{If } \beta = \beta' \textit{ then output } 1,\ \textit{and } 0\ \textit{otherwise.}
\end{aligned}$$

Here, $\mathcal{O} := \{\mathcal{O}_C(\cdot), \mathcal{O}_{Trap}(\cdot), \mathcal{O}_{Trans}(\cdot,\cdot)\}$. $\mathcal{O}_C$ takes $kw \in \mathcal{KS}$ as input, and returns the result of $\mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$. *Here, $kw \notin \{kw^*_0, kw^*_1\}$. $\mathcal{O}_{Trap}$ takes $kw' \in \mathcal{KS}$ as input, and returns the result of* $\mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw')$. *Here, $kw' \notin \{kw^*_0, kw^*_1\}$. $\mathcal{O}_{Trans}$ takes $\mathsf{ct_{DS\text{-}PAEKS}}$ and td_{kw}, and returns the result of* $\mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct_{DS\text{-}PAEKS}}, \mathsf{td}_{kw})$. *Here, there is no restriction.* DS-PAEKS *is IND-TS-IKGA secure for the test server if the advantage*

$$\mathsf{Adv}^{\mathsf{IND\text{-}TS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathsf{IND\text{-}TS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = 1] - 1/2|$$

is negligible in the security parameter λ for all PPT adversaries $\mathcal{A}$.

4 Vulnerability of Previous Schemes

4.1 Vulnerability of the Chen et al. DS-PAEKS scheme

The Chen et al. DS-PAEKS scheme [4] is described below:

$\mathsf{DS\text{-}PAEKS.Setup}(\lambda)$**:** The setup algorithm takes a security parameter λ as input, and outputs a common parameter $\mathsf{pp} = (\mathbb{G}, p, g_1, g_2, g_3, H)$, where $\mathbb{G}$ is a DDH-hard group with prime order p, $g_1, g_2, g_3 \in \mathbb{G}$ are distinct generators, and $H : \{0,1\} \to \mathbb{Z}_p$ is a collision-resistant hash function.

$\mathsf{DS\text{-}PAEKS.KG_R}(\mathsf{pp})$**:** Choose $d \xleftarrow{\$} \mathbb{Z}_p$. Output $\mathsf{pk_R} = g_3^d$ and $\mathsf{sk_R} = d$.

$\mathsf{DS\text{-}PAEKS.KG_S}(\mathsf{pp})$**:** Choose $c \xleftarrow{\$} \mathbb{Z}_p$. Output $\mathsf{pk_S} = g_3^c$ and $\mathsf{sk_S} = c$.

$\mathsf{DS\text{-}PAEKS.KG_{AS}}(\mathsf{pp})$**:** Choose $a \xleftarrow{\$} \mathbb{Z}_p$. Output $\mathsf{pk_{AS}} = g_1^a$ and $\mathsf{sk_{AS}} = a$.

$\mathsf{DS\text{-}PAEKS.KG_{TS}}(\mathsf{pp})$**:** Choose $b \xleftarrow{\$} \mathbb{Z}_p$. Output $\mathsf{pk_{TS}} = g_2^b$ and $\mathsf{sk_{TS}} = b$.

$\mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw)$**:** Choose $r_1 \xleftarrow{\$} \mathbb{Z}_p$. Compute $C_1 = g_1^{r_1}$, $C_2 = g_2^{r_1}$, and $C_3 = \mathsf{pk_{AS}}^{r_1}\mathsf{pk_{TS}}^{r_1}(\mathsf{pk_R}^{\mathsf{sk_S}})^{H(kw)}$ and output $\mathsf{ct_{DS\text{-}PAEKS}} = (C_1, C_2, C_3)$. Here, $C_3 = (g_1^a)^{r_1}(g_2^b)^{r_1}(g_3^{cd})^{H(kw)}$ holds.

$\mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_R}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw')$**:** Choose $r_2 \xleftarrow{\$} \mathbb{Z}_p$. Compute $T_1 = g_1^{r_2}$, $T_2 = g_2^{r_2}$, and $T_3 = \mathsf{pk_{AS}}^{r_2}\mathsf{pk_{TS}}^{r_2}/(\mathsf{pk_S}^{\mathsf{sk_R}})^{H(kw')}$, and output $\mathsf{td}_{kw'} = (T_1, T_2, T_3)$. Here, $T_3 = (g_1^a)^{r_2}(g_2^b)^{r_2}/(g_3^{cd})^{H(kw')}$ holds.

$\mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct_{DS\text{-}PAEKS}}, \mathsf{td}_{kw'})$**:** Parse $\mathsf{sk_{AS}} = a$, $\mathsf{ct_{DS\text{-}PAEKS}} = (C_1, C_2, C_3)$, and $\mathsf{td}_{kw'} = (T_1, T_2, T_3)$. Choose $r_3 \xleftarrow{\$} \mathbb{Z}_p$. Compute $ICT_1 = \{(C_3 \cdot T_3)/(C_1 \cdot T_1)^a\}^{r_3} = \{(g_1^a)^{r_1+r_2}(g_2^b)^{r_1+r_2}(g_3^{cd})^{H(kw)-H(kw')}/(g_1^{r_1+r_2})^a\}^{r_3} = (g_2^b)^{r_3(r_1+r_2)}(g_3^{cd})^{r_3(H(kw)-H(kw'))}$ and $ICT_2 = (C_2 \cdot T_2)^{r_3} = g_2^{r_3(r_1+r_2)}$. Output $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (ICT_1, ICT_2)$.

$\mathsf{DS\text{-}PAEKS.Test}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}}, \mathsf{int\text{-}ct_{DS\text{-}PAEKS}})$**:** Parse $\mathsf{sk_{TS}} = b$ and $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (ICT_1, ICT_2)$. Output 1 if $ICT_1 = ICT_2^b$ holds, and 0 otherwise.

Chen et al. claimed that $ICT_1 = (g_2^b)^{r_3(r_1+r_2)}(g_3^{cd})^{r_3(H(kw)-H(kw'))} = (g_2^b)^{r_3(r_1+r_2)} = (g_2^{r_3(r_1+r_2)})^b = ICT_2^b$ holds if $kw = kw'$. Due to the collision resistance of H, $H(kw) \neq H(kw')$ holds if $kw \neq kw'$. Thus, the $\mathsf{DS\text{-}PAEKS.Test}$ algorithm outputs 0 if $kw \neq kw'$.

Our attack is described here. The main problem is that the forms of ciphertext $\mathsf{ct_{DS\text{-}PAEKS}} = (C_1, C_2, C_3)$ and trapdoor $\mathsf{td}_{kw'}$ are almost the same, and an intermediate ciphertext $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ can be constructed from two ciphertexts (and $\mathsf{sk_{AS}} = a$) without using any trapdoor. Let $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}} = (C_1^*, C_2^*, C_3^*)$ be the challenge ciphertext where $C_1^* = g_1^{r_1^*}$, $C_2^* = g_2^{r_1^*}$, and $C_3^* = \mathsf{pk_{AS}}^{r_1^*}\mathsf{pk_{TS}}^{r_1^*}(\mathsf{pk_R}^{\mathsf{sk_S}})^{H(kw_\beta^*)}$ and $\beta \in \{0,1\}$. The IND-AS-CKA adversary that has $\mathsf{sk_{AS}} = a$ issues kw_0^* to the encryption oracle $\mathcal{O}_C$, and obtains $\mathsf{ct_{DS\text{-}PAEKS}} = (C_1, C_2, C_3)$ where $C_1 = g_1^{r_1}$, $C_2 = g_2^{r_1}$, and $C_3 = \mathsf{pk_{AS}}^{r_1}\mathsf{pk_{TS}}^{r_1}(\mathsf{pk_R}^{\mathsf{sk_S}})^{H(kw_0^*)}$. The adversary then prepares an intermediate ciphertext as follows:

- Choose $r_3 \xleftarrow{\$} \mathbb{Z}_p$.
- Compute $ICT_1 = \{(C_3^*/C_3)/(C_1^*/C_1)^a\}^{r_3}$ and $ICT_2 = (C_2^*/C_2)^{r_3}$. Here,

$$\begin{aligned} ICT_1 &= \{(C_3^*/C_3)/(C_1^*/C_1)^a\}^{r_3} \\ &= \{(g_1^a)^{r_1^*-r_1}(g_2^b)^{r_2^*-r_2}(g_3^{cd})^{H(kw_\beta^*)-H(kw_0^*)}/(g_1^{r_1^*-r_1})^a\}^{r_3} \\ &= (g_2^b)^{r_3(r_2^*-r_2)}(g_3^{cd})^{r_3(H(kw_\beta^*)-H(kw_0^*))} \\ ICT_2 &= (C_2^*/C_2)^{r_3} \\ &= g_2^{r_3(r_2^*-r_2)} \end{aligned}$$

  hold. The adversary sends $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (ICT_1, ICT_2)$ to the test oracle $\mathcal{O}_{\mathrm{Test}}$. If $\beta = 0$, then $ICT_1 = ICT_2^b$ holds and thus, the oracle outputs 1, and 0 otherwise. Thus, the adversary wins.

4.2 Vulnerability of the Chen et al. DS-PEKS scheme

The similar attack works against the Chen et al. DS-PEKS scheme [5].[2] The ciphertext form is $(g_1^{r_1}, g_2^{r_1}, h_1^{r_1}h_2^{r_1}H(kw))$ (now, no sender secret key is required for encryption). Here, the hash

[2]Here, we give an attack against the DDH-based construction given in [5]. However, our attack works against their generic construction from smooth projective hash functions.

function H is defined as $H : \{0,1\}^* \to \mathbb{G}$. In their security definition (SS-CKA: semantic-security against the chosen keyword attack, Fig. 1. in [5]), the oracle $\mathcal{O}_T$ is defined such that it takes a ciphertext and a keyword kw as input, and the oracle internally generates a trapdoor of kw and the intermediate ciphertext (internal testing state in [5]), and returns the result of the test algorithm. Here, $kw \notin \{kw_0^*, kw_1^*\}$ is required.[3] Thus, the same strategy as above does not work. However, because of the malleability of the ciphertext, we can modify the challenge ciphertext as follows. L Let $(C_1^*, C_2^*, C_3^*) = (g_1^{r_1^*}, g_2^{r_1^*}, h_1^{r_1^*} h_2^{r_1^*} H(kw_\beta^*))$ be the challenge ciphertext. The adversary computes $H(kw_0^*)$ and $H(kw)$ for arbitrary keyword $kw \notin \{kw_0^*, kw_1^*\}$. Then, the adversary chooses $r \xleftarrow{\$} \mathbb{Z}_p$ and computes $h_1^r h_2^r H(kw) C_3^* / H(kw_0^*) = H(kw) h_1^{r_1^*+r} h_2^{r_1^*+r} H(kw_\beta^*)/H(kw_0^*)$. If $\beta = 0$, then the ciphertext is an encryption of kw. If $\beta = 1$, then the ciphertext is an encryption of an unknown keyword (i.e., kw' where $H(kw') = H(kw)H(kw_1^*)/H(kw_0^*)$ holds). Here, kw is not equal to the unknown keyword because if the unknown keyword equals kw, then $H(kw)H(kw_1^*)/H(kw_0^*) = H(kw)$ holds and thus $H(kw_1^*) = H(kw_0^*)$. This contradicts the collision resistance of H because $kw_0^* \neq kw_1^*$. Thus, the adversary sends $(g_1^r C_1^*, g_2^r C_2^*, h_1^r h_2^r H(kw) C_3^*/H(kw_0^*)))$ and kw to $\mathcal{O}_T$. If the oracle returns 1, then $\beta = 0$, and $\beta = 1$ otherwise. Thus, the adversary wins.

4.3 Vulnerability of the Tso et al. DS-PEKS construction

Tso et al. [22] gave a semi-generic construction of DS-PEKS scheme from a PKE scheme. In their syntax, there are two servers, front server (assistant server in our notation) and back server (test server in our notation). Briefly, they employed a Pedersen commitment $g^r h^{kw}$ and encrypt X^r by using the underlying PKE scheme using the public key of the back server pk_{BS}, where $X = g^x$ is a public key of the front server. A ciphertext is described as $(g^r h^{kw}, \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^r))$. A trapdoor has a similar form: $(g^{r'} h^{-kw'}, \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r'}))$. The front server generates an intermediate ciphertext (they call it internal-testing-stage) using the secret key x such that $R((g^r h^{kw})(g^{r'} h^{-kw'}))^x = RX^{r+r'} h^{x(kw-kw')}$, where R is a random value. The intermediate ciphertext is described as $(\mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^r), \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r'}), H(R), RX^{r+r'} h^{x(kw-kw')})$ where H is a hash function. If $kw = kw'$, then it is described as $(\mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^r), \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r'}), H(R), RX^{r+r'})$. The back server decrypts $(\mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^r), \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r'}))$, obtains $(X^r, X^{r'})$, and checks $H(RX^{r+r'} h^{kw-kw'}/X^r X^{r'}) = H(R)$ holds or not. If it holds, then output 1, and 0 otherwise. They claimed that information about keyword is perfectly hidden by g^r and $g^{r'}$.

The main problem here is that the PKE part is independent of the keyword to be searched and the CCA security of the PKE scheme is meaningless to hide information about keyword. Actually, due to the homomorphic property of the commitment part, an adversary $\mathcal{A}$ can know $\beta = 0$ or $\beta = 1$ as follows. Here, $\mathcal{A}$ is modeled as a malicious back server that has the secret key of the PKE scheme sk_{BS} and the public key of the front server X (but $\mathcal{A}$ does not know the secret key of the front server x) (See the definition of IND-CKA-BS in [22]). Let the challenge ciphertext and the challenge trapdoor be described as $c_\beta = (g^r h^{kw_\beta^*}, \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^r))$ and $t_\beta = (g^{r'} h^{-kw_\beta^*}, \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r'}))$. Note that, $\mathcal{A}$ declares the challenge keywords (kw_0^*, kw_1^*), and the challenge ciphertext and the challenge trapdoor are given to the adversary simultaneously in their security model. $\mathcal{A}$ is allowed to access the front test oracle that takes a ciphertext $c \neq c_\beta$ and a trapdoor $t \neq t_\beta$, and returns the corresponding intermediate ciphertext. $\mathcal{A}$ prepares another ciphertext from c_β as follows. $\mathcal{A}$ decrypts $\mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^r)$ using sk_{BS} and obtains X^r. $\mathcal{A}$ randomly

[3]Tso et al. [22] have pointed out that the Chen et al. DS-PEKS scheme [5] is not as secure as they claimed. Basically, their attack is almost the same as ours, focusing on the linearity of smooth projective hash functions and using the test oracle. However, they generated another ciphertext of the challenge keyword from the challenge ciphertext, and sends the ciphertext and kw_1^* to the test oracle $\mathcal{O}_T$, that contradicts the restriction $kw \notin \{kw_0^*, kw_1^*\}$.

selects r'' and computes $g^{r''}g^r h^{kw_\beta^*} = g^{r''+r}h^{kw_\beta^*}$, $X^{r''}X^r = X^{r''+r}$, and $\mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r''+r})$. Now $c = (g^{r''+r}h^{kw_\beta^*}, \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r''+r}))$ is a ciphertext of kw_β^* and $c \neq c_\beta$. Then, $\mathcal{A}$ generates a trapdoor t for kw_0^* and then $t \neq t_\beta$. Let r''' be used as the randomness. $\mathcal{A}$ sends (c, t) to the front test oracle, and obtains the corresponding intermediate ciphertext. The intermediate ciphertext is described as $(\mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r''+r}), \mathsf{PKE.Enc}(\mathsf{pk}_{BS}, X^{r'''}), H(R), RX^{r''+r+r'''}h^{x(kw_\beta^*-kw_0^*)})$. If $H(RX^{r''+r+r'''}h^{x(kw_\beta^*-kw_0^*)}/X^{r''+r}X^{r'''}) = H(R)$, then $\beta = 0$, and $\beta = 1$ otherwise. Thus, the adversary wins.

4.4 Analysis of Other Pairing-free Schemes

Du et al. [11] and Lu and Li [16] proposed PAEKS schemes without pairings (in the designated-tester setting). Though we did not find any attack against the Du et al. scheme and the Lu-Li scheme, we show that at least their security proofs are wrong.

Du et al. scheme: They employed the hashed Diffie-Hellman (HDH) assumption: given (g, g^a, g^b, R), it is hard to decide $R = H(g^{ab})$ or not where H is a hash function. To generate the challenge ciphertext, $t = g^{H(kw||g^{ab}||g^a||g^b)}$ is computed. Du et al. randomly select R, compute g^R instead of computing g^t, and claim that this modification is indistinguishable if the HDH assumption holds. However, the simulation fails since $g^R = g^{H(g^{ab})}$ holds if $R = H(g^{ab})$ and this does not appropriately simulate g^t.

Lu-Li scheme: They employed the DDH assumption: given (g, g^a, g^b, R), it is hard to decide $R = g^{ab}$ or not. In their security proof, two challenge users, say I and J, are selected and their keys are set as $PK_I = (PK_{I,1}, PK_{I,2}) := (g^{x_I}, g^a)$ and $PK_J = (PK_{J,1}, PK_{J,2}) := (g^{x_J}, g^b)$ where x_I and x_J are chosen by the simulator and g^a and g^b are the DDH instance that a and b are unknown. A ciphertext of kw generated by $SK_I = (x_I, a)$ and PK_J consists of $IC_1 = g^r$ and $IC_2 = H_3(Q)$ where $Q = (gPK_{J,2}^{H_2(kw,\lambda_1,\lambda_2)})^r$, $\lambda_1 = H_1(PK_{I,1}, PK_{J,1}, (PK_{J,1})^{x_I})$, and $\lambda_2 = H_1(PK_{I,2}, PK_{J,2}, (PK_{J,2})^a)$. An adversary is allowed to issue a ciphertext query (PK_I, PK_J, kw) if $kw \notin \{kw_0^*, kw_1^*\}$. Here, H_1, H_2, and H_3 are hash functions modeled as random oracles. To respond to the ciphertext query, the simulator needs to compute λ_2 that requires to compute $PK_{J,2}^a = g^{ab}$. However, this requires to solve the computational Diffie-Hellman problem: given (g, g^a, g^b), compute g^{ab}. Thus, the simulation fails. In the security proof, it is assumed that no (g^a, g^b, S) is queried to H_1 where (g, g^a, g^b, S) is a valid DDH tuple. However, the adversary can make the query via the ciphertext oracle as above.

As another problem, for $PK = (PK_1, PK_2) = (g^{a_1}, g^{a_2})$, $SK_1 = a_1$ is extracted by an adversary $\mathcal{A}$ though $\mathcal{A}$ did not send a corruption query for (PK_1, PK_2). In their scheme, a trapdoor is $td = SK_1 H_2(kw, \lambda_1, \lambda_2)$. First, $\mathcal{A}$ issues a corruption query $PK' = (PK_1', PK_2')$, obtains (SK_1', SK_2'), and issues a trapdoor query (kw, PK', PK). Then the oracle responds $SK_1 H_2(kw, \lambda_1, \lambda_2)$ where $\lambda_1 = H_1(PK, PK', (PK_1')^{SK_1}) = H_1(PK, PK', PK_1^{SK_1'})$ and $\lambda_2 = H_1(PK, PK', (PK_2')^{SK_2}) = H_1(PK, PK', PK_2^{SK_2'})$. Since A knows (SK_1', SK_2'), $\mathcal{A}$ can compute λ_1 and λ_2. Thus, from the trapdoor, $\mathcal{A}$ can compute $SK_1 = td/H_2(kw, \lambda_1, \lambda_2)$. Since both secret keys are required to compute a trapdoor, the situation revealing SK_1 does not immediately break the scheme, i.e., still $\mathcal{A}$ is not able to generate a trapdoor that works to distinguish whether the challenge ciphertext is an encryption of kw_0^* or kw_1^*. Nevertheless, the scheme structure should be reconsidered because revealing a part of secret key without corruption is a fatal error as a cryptographic primitive.

5 Proposed Generic construction

Technical Overview: The proposed generic construction employs Chen's idea [6], i.e., when a server has a public key, and the test algorithm takes the secret key of the server as input, then it is sufficient to encrypt a trapdoor by the public key to hide information about the keywords associated with the trapdoor. In our construction, the assistant and test servers manage public keys of PKE, respectively. A sender re-encrypts a PAEKS ciphertext using $\mathsf{pk}_{\mathsf{AS}}$, and sends it to the assistant server as a DS-PAEKS ciphertext. A receiver encrypts a PAEKS trapdoor using $\mathsf{pk}_{\mathsf{TS}}$, and sends it to the assistant server as a DS-PAEKS trapdoor. The assistant server then decrypts the DS-PAEKS ciphertext using $\mathsf{sk}_{\mathsf{AS}}$, and sets the PAEKS ciphertext and the DS-PAEKS trapdoor as the intermediate ciphertext. The test server decrypts the DS-PAEKS trapdoor using $\mathsf{sk}_{\mathsf{TS}}$, and obtains the PAEKS trapdoor. The test server then runs the test algorithm of the underlying PAEKS scheme. The construction is basically secure because no information about keywords is leaked from PAEKS ciphertexts and PAEKS trapdoors due to the security of the underlying PAEKS scheme. Moreover, no single server can run the PAEKS test algorithm because either a PAEKS ciphertext or a PAEKS trapdoor is encrypted using the public key of the other server. We also introduce two signature schemes, where a sender and a receiver sign a PAEKS ciphertext and a PAEKS trapdoor, respectively, before the encryption to exclude the case of an adversary producing a PKE ciphertext of self-made PAEKS ciphertexts/trapdoors for the challenge keyword.

Let $\mathsf{PAEKS} = (\mathsf{PAEKS.Setup}, \mathsf{PAEKS.KG}_{\mathsf{R}}, \mathsf{PAEKS.KG}_{\mathsf{S}}, \mathsf{PAEKS.Enc}, \mathsf{PAEKS.Trapdoor}, \mathsf{PAEKS.Test})$ be a PAEKS scheme, $\mathsf{PKE} = (\mathsf{PKE.KeyGen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$ be a PKE scheme, and $\mathsf{Sig} = (\mathsf{Sig.KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ be a signature scheme. We construct a DS-PAEKS scheme $\mathsf{DS\text{-}PAEKS} = (\mathsf{DS\text{-}PAEKS.Setup}, \mathsf{DS\text{-}PAEKS.KG}_{\mathsf{R}}, \mathsf{DS\text{-}PAEKS.KG}_{\mathsf{S}}, \mathsf{DS\text{-}PAEKS.KG}_{\mathsf{AS}}, \mathsf{DS\text{-}PAEKS.KG}_{\mathsf{TS}}, \mathsf{DS\text{-}PAEKS.Enc}, \mathsf{DS\text{-}PAEKS.Trapdoor}, \mathsf{DS\text{-}PAEKS.Transition}, \mathsf{DS\text{-}PAEKS.Test})$ as follows.

$\mathsf{DS\text{-}PAEKS.Setup}(\lambda)$**:** Run $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$ and output pp. We assume that pp contains the security parameter λ.

$\mathsf{DS\text{-}PAEKS.KG}_{\mathsf{R}}(\mathsf{pp})$**:** Run $(\mathsf{pk}'_{\mathsf{R}}, \mathsf{sk}'_{\mathsf{R}}) \leftarrow \mathsf{PAEKS.KG}_{\mathsf{R}}(\mathsf{pp})$ and $(\mathsf{vk}_{\mathsf{R}}, \mathsf{sigk}_{\mathsf{R}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. Output $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$ and $\mathsf{sk}_{\mathsf{R}} = (\mathsf{sk}'_{\mathsf{R}}, \mathsf{sigk}_{\mathsf{R}})$.

$\mathsf{DS\text{-}PAEKS.KG}_{\mathsf{S}}(\mathsf{pp})$**:** Run $(\mathsf{pk}'_{\mathsf{S}}, \mathsf{sk}'_{\mathsf{S}}) \leftarrow \mathsf{PAEKS.KG}_{\mathsf{S}}(\mathsf{pp})$ and $(\mathsf{vk}_{\mathsf{S}}, \mathsf{sigk}_{\mathsf{S}}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. Output $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$ and $\mathsf{sk}_{\mathsf{S}} = (\mathsf{sk}'_{\mathsf{S}}, \mathsf{sigk}_{\mathsf{S}})$.

$\mathsf{DS\text{-}PAEKS.KG}_{\mathsf{AS}}(\mathsf{pp})$**:** Run $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and output $\mathsf{pk}_{\mathsf{AS}} = \mathsf{PK}$ and $\mathsf{sk}_{\mathsf{AS}} = \mathsf{DK}$.

$\mathsf{DS\text{-}PAEKS.KG}_{\mathsf{TS}}(\mathsf{pp})$**:** Run $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and output $\mathsf{pk}_{\mathsf{TS}} = \mathsf{PK}'$ and $\mathsf{sk}_{\mathsf{TS}} = \mathsf{DK}'$.

$\mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{AS}}, \mathsf{pk}_{\mathsf{TS}}, kw)$**:** Parse $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$, $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$, and $\mathsf{sk}_{\mathsf{S}} = (\mathsf{sk}'_{\mathsf{S}}, \mathsf{sigk}_{\mathsf{S}})$. Run $\mathsf{ct}_{\mathsf{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}'_{\mathsf{R}}, \mathsf{pk}'_{\mathsf{S}}, \mathsf{sk}'_{\mathsf{S}}, kw)$, $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}_{\mathsf{S}}, \mathsf{ct}_{\mathsf{PAEKS}})$, and $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{AS}}, \sigma || \mathsf{ct}_{\mathsf{PAEKS}})$. Output $\mathsf{ct}_{\mathsf{DS\text{-}PAEKS}} = C$.

$\mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{sk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{AS}}, \mathsf{pk}_{\mathsf{TS}}, kw')$**:** Parse $\mathsf{pk}_{\mathsf{R}} = (\mathsf{pk}'_{\mathsf{R}}, \mathsf{vk}_{\mathsf{R}})$, $\mathsf{sk}_{\mathsf{R}} = (\mathsf{sk}'_{\mathsf{R}}, \mathsf{sigk}_{\mathsf{R}})$, and $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$. Run $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_{\mathsf{R}}, \mathsf{pk}'_{\mathsf{S}}, \mathsf{sk}'_{\mathsf{R}}, kw')$, $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk}_{\mathsf{R}}, \mathsf{td}'_{kw'})$, and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_{\mathsf{TS}}, \sigma' || \mathsf{td}'_{kw'})$. Output $\mathsf{td}_{kw'} = C'$.

$\mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk}_{\mathsf{R}}, \mathsf{pk}_{\mathsf{S}}, \mathsf{pk}_{\mathsf{AS}}, \mathsf{sk}_{\mathsf{AS}}, \mathsf{ct}_{\mathsf{DS\text{-}PAEKS}}, \mathsf{td}_{kw'})$**:** Parse $\mathsf{pk}_{\mathsf{S}} = (\mathsf{pk}'_{\mathsf{S}}, \mathsf{vk}_{\mathsf{S}})$ and $\mathsf{ct}_{\mathsf{DS\text{-}PAEKS}} = C$. Run $\sigma || \mathsf{ct}_{\mathsf{PAEKS}} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_{\mathsf{AS}}, C)$. Output $\perp$ if $\mathsf{Verify}(\mathsf{vk}_{\mathsf{S}}, \mathsf{ct}_{\mathsf{PAEKS}}, \sigma) = 0$. Otherwise, output $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}} = (\mathsf{ct}_{\mathsf{PAEKS}}, \sigma, \mathsf{td}_{kw'})$.

$\mathsf{DS\text{-}PAEKS.Test}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}}, \mathsf{int\text{-}ct_{DS\text{-}PAEKS}})$**:** Parse $\mathsf{pk_R} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk_R})$, $\mathsf{pk_S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk_S})$, and $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ where $\mathsf{td}_{kw'} = C'$. Output 0 if $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{ct_{PAEKS}}, \sigma) = 0$. Otherwise, run $\sigma'||\mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk_{TS}}, C')$. Output 0 if $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 0$. Otherwise, output the result of $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw'})$.

The proposed construction is correct if the underlying PAEKS, PKE, and signature schemes are correct. Moreover, the proposed construction is computationally consistent if the underlying PAEKS scheme is computationally consistent. We note that signature schemes are related to the result of the DS-PAEKS.Test algorithm. However, they are employed for preventing any modification of the challenge ciphertext and trapdoor. Thus, the proposed construction provides computational consistency even if signature schemes are insecure (e.g., the Verify algorithm always outputs 1 regardless of the input). Precisely, if the DS-PAEKS.Test algorithm outputs 1, then the PAEKS.Test algorithm must output 1, and the result of the Verify algorithm is independent.

6 Security Analysis

Theorem 1. *The proposed construction is IND-AS-CKA secure if* PAEKS *is IND-CKA secure,* PKE *is IND-CCA secure, and* Sig *is sEUF-CMA secure.*

Basically, the IND-AS-CKA security is reduced to the IND-CKA security of PAEKS. However, we must consider two main cases: (1) how to simulate $\mathcal{O}_{\mathrm{Trap}}$ for $kw \in \{kw_0^*, kw_1^*\}$ because $\mathcal{O}_{\mathrm{Trap}}$ of the underlying PAEKS scheme has the restriction that $kw \notin \{kw_0^*, kw_1^*\}$, and (2) how to prevent any modification of trapdoors of the challenge keyword because if an adversary issues $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}}$ to $\mathcal{O}_{\mathrm{Test}}$, where either $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}}, \mathsf{td}_{kw_0^*})$ $\wedge\ \mathsf{td}_{kw_0^*} \notin \mathsf{TSet}$ or $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}}, \mathsf{td}_{kw_1^*})$ $\wedge$ $\mathsf{td}_{kw_1^*} \notin \mathsf{TSet}$, then the adversary trivially wins. We handled the first issue by employing the IND-CPA security of the underlying PKE scheme. PAEKS trapdoors are now encrypted by the public key of the test server. Thus, the PKE ciphertext of the PAEKS trapdoor for $kw \in \{kw_0^*, kw_1^*\}$ can be replaced with a PKE ciphertext of 0 due to the IND-CPA security. Then, the simulator does not have to issue a trapdoor query to the underlying PAEKS scheme. More precisely, we require that the underlying PKE scheme is IND-CCA secure to simulate $\mathcal{O}_{\mathrm{Test}}$ that internally runs the decryption algorithm of PKE. We handled the second issue by employing the sEUF-CMA security of the underlying signature scheme. That is, a PAEKS trapdoor is signed before encryption to prevent any PAEKS trapdoor modification. One may think that the signature scheme is redundant because the PAEKS trapdoors are encrypted by the IND-CCA secure PKE that prevents the PKE ciphertext modification. However, we must exclude the case in which an adversary produces a PKE ciphertext of a self-made PAEKS trapdoor for the challenge keyword. Thus, we employ both the PKE and signature schemes in the proposed construction.

Proof. The proof uses a sequence of games. Let E_i be the event in which $\mathcal{A}$ outputs $\beta' = \beta$ in Game i.

Game 0: This game corresponds to the real game. By definition, $\mathsf{Adv}^{\mathsf{IND\text{-}AS\text{-}CKA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = |\Pr[E_0] - 1/2|$.

Game 1: This game is the same as Game 0, except that the response of the $\mathcal{O}_{\mathrm{Test}}$ oracle is changed as follows. Let $\mathcal{A}$ be an IND-AS-CKA adversary. Assume that $\mathcal{A}$ issues $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ to $\mathcal{O}_{\mathrm{Test}}$. Run $\sigma'||\mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk_{TS}}, \mathsf{td}_{kw'})$. If $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 1$ and $(\mathsf{td}'_{kw'}, \sigma')$ was not generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle, then the challenger aborts. If the challenger does not abort, then Game 1 is identical to Game 0. Thus, $|\Pr[E_0] - \Pr[E_1]| \leq \Pr[\mathsf{abort}]$ where abort is the event that the challenger aborts.

Lemma 1. *There exists an algorithm $\mathcal{B}$ such that* $\Pr[\mathsf{abort}] \leq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}}(\lambda)$.

Proof. Let $\mathcal{A}$ be the adversary of IND-AS-CKA and $\mathcal{C}$ be the challenger of the signature scheme. We construct an algorithm $\mathcal{B}$ that breaks the sEUF-CMA security as follows. First, $\mathcal{B}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{R}, \mathsf{sk}'_\mathsf{R}) \leftarrow \mathsf{PAEKS.KG}_\mathsf{R}(\mathsf{pp})$, $(\mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}) \leftarrow \mathsf{PAEKS.KG}_\mathsf{S}(\mathsf{pp})$, $(\mathsf{vk}_\mathsf{S}, \mathsf{sigk}_\mathsf{S}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$. $\mathcal{C}$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, and sends vk to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk}_\mathsf{R} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk})$, $\mathsf{sk}_\mathsf{R} = (\mathsf{sk}'_\mathsf{R}, -)$, $\mathsf{pk}_\mathsf{S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk}_\mathsf{S})$, $\mathsf{sk}_\mathsf{S} = (\mathsf{sk}'_\mathsf{S}, \mathsf{sigk}_\mathsf{S})$, $\mathsf{pk}_\mathsf{AS} = \mathsf{PK}$, $\mathsf{sk}_\mathsf{AS} = \mathsf{DK}$, $\mathsf{pk}_\mathsf{TS} = \mathsf{PK}'$, and $\mathsf{sk}_\mathsf{TS} = \mathsf{DK}'$, and sends $(\mathsf{pp}, \mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{AS}, \mathsf{sk}_\mathsf{AS}, \mathsf{pk}_\mathsf{TS})$ to $\mathcal{A}$. $\mathcal{B}$ sets $\mathsf{TSet} := \emptyset$ and $\mathsf{SSet} := \emptyset$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to any query because $\mathcal{B}$ has sk_S.

- For $\mathcal{O}_{\mathrm{Trap}}$, $\mathcal{B}$ can respond to a query kw' from $\mathcal{A}$ as follows. $\mathcal{B}$ runs $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw')$ and sends $\mathsf{td}'_{kw'}$ to $\mathcal{C}$ as a signing query. $\mathcal{C}$ returns $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk}, \mathsf{td}'_{kw'})$ to $\mathcal{B}$. $\mathcal{B}$ computes $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{TS}, \sigma'||\mathsf{td}'_{kw'})$ and returns $\mathsf{td}_{kw'} = C'$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ stores $(\mathsf{td}'_{kw'}, \sigma')$ on SSet.

- For $\mathcal{O}_{\mathrm{Test}}$, $\mathcal{B}$ can respond to a query $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}} = (\mathsf{ct}_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw'})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk}_\mathsf{S}, \mathsf{ct}_\mathsf{PAEKS}, \sigma) = 0$. Otherwise, $\mathcal{B}$ runs $\sigma'||\mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\mathsf{TS}, \mathsf{td}_{kw'})$. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk}_\mathsf{R}, \mathsf{td}'_{kw'}, \sigma') = 0$. From now on, we assume that $\mathsf{Verify}(\mathsf{vk}_\mathsf{R}, \mathsf{td}'_{kw'}, \sigma') = 1$. If $(\mathsf{td}'_{kw'}, \sigma')$ is generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle, then $\mathcal{B}$ returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct}_\mathsf{PAEKS}, \mathsf{td}'_{kw'})$. If $(\mathsf{td}'_{kw'}, \sigma')$ was not generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle, i.e., $(\mathsf{td}'_{kw'}, \sigma') \notin \mathsf{SSet}$, then σ' is not a response from $\mathcal{C}$. Thus, $\mathcal{B}$ outputs $(\mathsf{td}'_{kw'}, \sigma')$ as a forged message and signature pair, and breaks the sEUF-CMA security of the signature scheme.

In the challenge phase, $\mathcal{A}$ declares (kw_0^*, kw_1^*). $\mathcal{B}$ chooses $\beta \xleftarrow{\$} \{0,1\}$, generates $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{S}, \mathsf{pk}_\mathsf{AS}, \mathsf{pk}_\mathsf{TS}, kw^*_\beta)$, and returns $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}}$ to $\mathcal{A}$.

$\mathcal{B}$ simulates $\mathcal{O}_C$, $\mathcal{O}_{\mathrm{Trap}}$, and $\mathcal{O}_{\mathrm{Test}}$ as in the previous phase, except that if $\mathcal{A}$ sends $kw \in \{kw_0^*, kw_1^*\}$ to $\mathcal{O}_{\mathrm{Trap}}$, then $\mathcal{B}$ updates $\mathsf{TSet} = \mathsf{TSet} \cup \{\mathsf{td}_{kw}\}$ where td_{kw} is the response of the $\mathcal{O}_{\mathrm{Trap}}$ oracle. If $\mathcal{A}$ does not issue a test query $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}} = (\mathsf{ct}_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw'})$ where, for $\sigma'||\mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\mathsf{TS}, \mathsf{td}_{kw'})$, $\mathsf{Verify}(\mathsf{vk}_\mathsf{R}, \mathsf{td}'_{kw'}, \sigma') = 1$ and $(\mathsf{td}'_{kw'}, \sigma')$ was not generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle, then $\mathcal{B}$ simulates Game 0, and Game 1 otherwise. □

Game $2.k$ $(1 \leq k \leq q_\mathsf{Trap})$**:** Let q_Trap be the number of trapdoor queries issued by $\mathcal{A}$ and Game 2.0 be the same as Game 1. Game $2.k$ is the same as Game $2.k-1$, except that the response of the k-th $\mathcal{O}_{\mathrm{Trap}}$ query is changed as follows. Let $\mathcal{A}$ issues kw to $\mathcal{O}_{\mathrm{Trap}}$ as the k-th query. Let ℓ be the bit size of the PAEKS trapdoor. Set $\mathsf{td}'_{kw} = 0^{|\ell|}$ and run $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{R}, \mathsf{td}'_{kw'})$, and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{TS}, \sigma'||\mathsf{td}'_{kw'})$. Output $\mathsf{td}_{kw'} = C'$.

Lemma 2. *For each $k \in [1, q_\mathsf{Trap}]$, Game $2.k$ is indistinguishable from Game $2.k-1$ if the underlying PKE scheme is IND-CCA secure. Precisely, there exists an algorithm $\mathcal{B}$ such that* $|\Pr[E_{2.k-1}] - \Pr[E_{2.k}]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$.

Proof. Let $\mathcal{A}$ be the adversary of IND-AS-CKA and $\mathcal{C}$ be the challenger of the PKE scheme. We construct an algorithm $\mathcal{B}$ that breaks the IND-CCA security as follows. $\mathcal{C}$ runs $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and sends PK' to $\mathcal{B}$. $\mathcal{B}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{R}, \mathsf{sk}'_\mathsf{R}) \leftarrow \mathsf{PAEKS.KG}_\mathsf{R}(\mathsf{pp})$, $(\mathsf{vk}_\mathsf{R}, \mathsf{sigk}_\mathsf{R}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}) \leftarrow \mathsf{PAEKS.KG}_\mathsf{S}(\mathsf{pp})$, $(\mathsf{vk}_\mathsf{S}, \mathsf{sigk}_\mathsf{S}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, and

$(\mathsf{PK},\mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$. $\mathcal{B}$ sets $\mathsf{pk_{TS}} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk})$, $\mathsf{sk_R} = (\mathsf{sk}'_\mathsf{R}, \mathsf{sigk_R})$, $\mathsf{pk_S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk_S})$, $\mathsf{sk_S} = (\mathsf{sk}'_\mathsf{S}, \mathsf{sigk_S})$, $\mathsf{pk_{AS}} = \mathsf{PK}$, $\mathsf{sk_{AS}} = \mathsf{DK}$, $\mathsf{pk_{TS}} = \mathsf{PK}'$, and $\mathsf{sk_{TS}} = -$, and sends $(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{pk_{TS}})$ to $\mathcal{A}$. $\mathcal{B}$ sets $\mathsf{TSet} := \emptyset$, $\mathsf{SSet} := \emptyset$, and $\mathsf{CSet} := \emptyset$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to any query because $\mathcal{B}$ has $\mathsf{sk_S}$.
- For $\mathcal{O}_{\mathrm{Trap}}$, $\mathcal{B}$ can respond to a query kw' from $\mathcal{A}$ as follows.
  - From 1 to $k-1$-th queries, $\mathcal{B}$ sets $\mathsf{td}'_{kw'} = 0^{|\ell|}$, computes $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, \mathsf{td}'_{kw'})$ and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{TS}}, \sigma'||\mathsf{td}'_{kw'})$, and returns $\mathsf{td}_{kw'} = C'$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ stores $(kw', \mathsf{td}'_{kw'}, \sigma')$ on SSet.
  - From $k+1$ to q_Trap queries, $\mathcal{B}$ runs $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw')$, $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, \mathsf{td}'_{kw'})$, and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{TS}}, \sigma'||\mathsf{td}'_{kw'})$, and returns $\mathsf{td}_{kw'} = C'$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ stores $(kw', \mathsf{td}'_{kw'}, \sigma')$ on SSet.
  - For the k-th query, $\mathcal{B}$ runs $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw')$ and computes $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, \mathsf{td}'_{kw'})$ and $\sigma'' \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, 0^{|\ell|})$. $\mathcal{B}$ sets $(\sigma'||\mathsf{td}'_{kw'}, \sigma''||0^{|\ell|})$ as the challenge plaintexts, and sends $(\sigma'||\mathsf{td}'_{kw'}, \sigma''||0^{|\ell|})$ to $\mathcal{C}$. $\mathcal{C}$ returns the challenge ciphertext C^*. $\mathcal{B}$ returns $\mathsf{td}_{kw'} = C^*$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ stores (kw', C^*) on CSet.
- For $\mathcal{O}_{\mathrm{Test}}$, $\mathcal{B}$ can respond to a query $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{ct_{PAEKS}}, \sigma) = 0$. If $\mathsf{td}_{kw'} = C^*$, then $\mathcal{B}$ knows the keyword associated to $\mathsf{td}_{kw'}$ because (kw', C^*) is stored on CSet. $\mathcal{B}$ runs $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}$ $(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw')$, and returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw'})$. If $\mathsf{td}_{kw'} \neq C^*$, then $\mathcal{B}$ sends $\mathsf{td}_{kw'}$ to $\mathcal{C}$ as a decryption query. If $\mathcal{C}$ returns $\perp$, then $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, let $\sigma'||\mathsf{td}'_{kw'}$ be the response from $\mathcal{C}$. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 0$. Otherwise, $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 1$. Due to the modification of Game 1, $(\mathsf{td}'_{kw'}, \sigma')$ was generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle. Thus, $(kw', \mathsf{td}'_{kw'}, \sigma') \in \mathsf{SSet}$. If $\mathsf{td}'_{kw'} = 0^{|\ell|}$, then $\mathcal{B}$ runs $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw')$. $\mathcal{B}$ returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw'})$.

In the challenge phase, $\mathcal{A}$ declares (kw_0^*, kw_1^*). $\mathcal{B}$ chooses $\beta \xleftarrow{\$} \{0,1\}$, generates $\mathsf{ct}^*_\mathsf{DS\text{-}PAEKS} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw^*_\beta)$, and returns $\mathsf{ct}^*_\mathsf{DS\text{-}PAEKS}$ to $\mathcal{A}$.

$\mathcal{B}$ simulates $\mathcal{O}_C$, $\mathcal{O}_{\mathrm{Trap}}$, and $\mathcal{O}_{\mathrm{Test}}$ as in the previous phase, except that if $\mathcal{A}$ sends $kw \in \{kw_0^*, kw_1^*\}$ to $\mathcal{O}_{\mathrm{Trap}}$, then $\mathcal{B}$ updates $\mathsf{TSet} = \mathsf{TSet} \cup \{\mathsf{td}_{kw}\}$ where td_{kw} is the response of the $\mathcal{O}_{\mathrm{Trap}}$ oracle. If the challenge ciphertext C^* is an encryption of $\sigma'||\mathsf{td}'_{kw}$, then $\mathcal{B}$ simulates Game $2.k-1$, and if C^* is an encryption of $\sigma''||0^{|\ell|}$, then $\mathcal{B}$ simulates Game $2.k$. Thus, $|\Pr[E_{2.k-1}] - \Pr[E_{2.k}]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$ holds. □

Game 3: This game is the same as Game $2.q_\mathsf{Trap}$, except that the response of $\mathcal{O}_{\mathrm{Test}}$ is changed as follows. If $\mathcal{A}$ issues $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw})$ such that (1) $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{ct_{PAEKS}}, \sigma) = 1$, (2) for $\sigma'||\mathsf{td}'_{kw} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk_{TS}}, \mathsf{td}_{kw})$, $(\mathsf{td}'_{kw}, \sigma')$ was generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle for a query kw (and thus $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 1$ and $(kw, \mathsf{td}'_{kw}, \sigma') \in \mathsf{SSet}$), (3) $kw \in \{kw_0^*, kw_1^*\}$, and (4) $\mathsf{td}_{kw} \notin \mathsf{TSet}$, then the challenger aborts. If challenger does not abort, then Game 3 is identical to Game $2.q_\mathsf{Trap}$. Thus, $|\Pr[E_{2.q_\mathsf{Trap}}] - \Pr[E_3]| \leq \Pr[\mathsf{abort}]$ where abort is the event when the challenger aborts.

Lemma 3. *There exists an algorithm* $\mathcal{B}$ *such that* $\Pr[\mathsf{abort}] \leq q_\mathsf{Trap} \cdot \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$.

Proof. Due to the modification in Game 1, $(kw, \mathsf{td}'_{kw}, \sigma') \in \mathsf{SSet}$. Thus, $kw \in \{kw_0^*, kw_1^*\}$ and $\mathsf{td}_{kw} \notin \mathsf{TSet}$ mean that a PKE ciphertext $\mathsf{PKE.Enc}(\mathsf{pk_{TS}}, \sigma'||\mathsf{td}'_{kw})$ is re-randomized by $\mathcal{A}$ that contradicts the IND-CCA security. Let $\mathcal{A}$ be the adversary of IND-AS-CKA and $\mathcal{C}$ be the challenger of the PKE scheme.

We construct an algorithm $\mathcal{B}$ that breaks the IND-CCA security as follows. $\mathcal{C}$ runs $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and sends PK' to $\mathcal{B}$. $\mathcal{B}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{R}, \mathsf{sk}'_\mathsf{R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp})$, $(\mathsf{vk_R}, \mathsf{sigk_R}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})$, $(\mathsf{vk_S}, \mathsf{sigk_S}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, and $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$. $\mathcal{B}$ sets $\mathsf{pk_{TS}} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk})$, $\mathsf{sk_R} = (\mathsf{sk}'_\mathsf{R}, \mathsf{sigk_R})$, $\mathsf{pk_S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk_S})$, $\mathsf{sk_S} = (\mathsf{sk}'_\mathsf{S}, \mathsf{sigk_S})$, $\mathsf{pk_{AS}} = \mathsf{PK}$, $\mathsf{sk_{AS}} = \mathsf{DK}$, $\mathsf{pk_{TS}} = \mathsf{PK}'$, and $\mathsf{sk_{TS}} = -$, and sends $(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{pk_{TS}})$ to $\mathcal{A}$. $\mathcal{B}$ sets $\mathsf{TSet} := \emptyset$ and $\mathsf{SSet} := \emptyset$. We assume that $(\mathsf{td}'_{kw}, \sigma')$ (appeared in the condition (2)) was generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle for the k^*-th trapdoor query. $\mathcal{B}$ guesses k^*. From now on, we assume that the guessing of k^* is correct (with the probability at least $1/q_{\mathsf{Trap}}$).

In the challenge phase, $\mathcal{A}$ declares (kw_0^*, kw_1^*). $\mathcal{B}$ chooses $\beta \xleftarrow{\$} \{0,1\}$, generates $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Enc}(\mathsf{pk_R}, \mathsf{pk_S}, \mathsf{sk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, kw_\beta^*)$, and returns $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}}$ to $\mathcal{A}$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to any query because $\mathcal{B}$ has $\mathsf{sk_S}$.
- $\mathcal{B}$ simulates $\mathcal{O}_{\mathrm{Trap}}$ oracle as in Game $2.q_{\mathsf{Trap}}$, except the k^*-th query. $\mathcal{B}$ simulates $\mathcal{O}_{\mathrm{Trap}}$ for the k^*-th query as follows. We remark that the k^*-th query may be sent from $\mathcal{A}$ before the challenge phase. We also remark that, for a trapdoor query kw, $\mathcal{B}$ sets $\mathsf{td}'_{kw} = 0^{|\ell|}$ regardless of the associated keyword due to the modification of previous games. $\mathcal{B}$ computes $\sigma'_0 \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, 0^{|\ell|})$ and $\sigma'_1 \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, 0^{|\ell|})$, and sets $(\sigma'_0||0^{|\ell|}, \sigma'_1||0^{|\ell|})$ as the challenge plaintexts. We remark that we have explicitly assumed that the Sign algorithm is probabilistic and thus $\sigma'_0 \neq \sigma'_1$ holds with overwhelming probability. $\mathcal{C}$ returns the challenge ciphertext $C^* \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{TS}}, \sigma'_{\beta'}||0^{|\ell|})$ where $\beta' \in \{0,1\}$. $\mathcal{B}$ returns $\mathsf{td}_{kw} = C^*$ to $\mathcal{A}$. $\mathcal{B}$ stores $(kw, 0^{|\ell|}, \sigma'_0, \sigma'_1)$ to SSet (here, the 4-th entry is added to store two signatures).
- For $\mathcal{O}_{\mathrm{Test}}$, $\mathcal{B}$ can respond to a query $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{ct_{PAEKS}}, \sigma) = 0$. If $\mathsf{td}_{kw'} = C^*$, then $\mathcal{B}$ knows the keyword associated to $\mathsf{td}_{kw'}$ because $(kw, 0^{|\ell|}, \sigma'_0, \sigma'_1)$ is stored on SSet. $\mathcal{B}$ runs $\mathsf{td}'_{kw} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw)$, and returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw})$. If $\mathsf{td}_{kw'} \neq C^*$, then $\mathcal{B}$ sends $\mathsf{td}_{kw'}$ to $\mathcal{C}$ as a decryption query. If $\mathcal{C}$ returns $\perp$, then $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, let $\sigma'||\mathsf{td}'_{kw'}$ be the response from $\mathcal{C}$. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 0$. Otherwise, $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 1$. Due to the modification of Game 1, $(\mathsf{td}'_{kw'}, \sigma')$ was generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle. If $(\mathsf{td}'_{kw'}, \sigma')$ was generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle for the k^*-th query, then $\mathsf{td}'_{kw} = 0^{|\ell|}$, $\sigma' \in \{\sigma'_0, \sigma'_1\}$ where $(kw, 0^{|\ell|}, \sigma'_0, \sigma'_1) \in \mathsf{SSet}$, $kw \in \{kw_0^*, kw_1^*\}$, and $\mathsf{td}_{kw} \notin \mathsf{TSet}$. $\mathcal{B}$ outputs 0 if $\sigma' = \sigma'_0$ and 1 if $\sigma' = \sigma'_1$ and breaks the IND-CCA security. If $(\mathsf{td}'_{kw'}, \sigma')$ was generated in the $\mathcal{O}_{\mathrm{Trap}}$ oracle for the k-th query where $k \neq k^*$, let $(kw', 0^{|\ell|}, \sigma') \in \mathsf{SSet}$. $\mathcal{B}$ runs $\mathsf{td}'_{kw'} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw')$ and returns the result of $\mathsf{PAEKS.Test}$ $(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw'})$. □

Lemma 4. *There exists an algorithm $\mathcal{B}$ such that* $|\Pr[E_3] - 1/2| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{B}}(\lambda)$.

Proof. Let $\mathcal{A}$ be the adversary of IND-AS-CKA and $\mathcal{C}$ be the challenger of the PAEKS scheme. We construct an algorithm $\mathcal{B}$ that breaks the IND-CKA security as follows. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{R}, \mathsf{sk}'_\mathsf{R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp})$, and $(\mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})$, and sends $(\mathsf{pp}, \mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S})$ to $\mathcal{B}$. $\mathcal{B}$ runs $(\mathsf{vk_R}, \mathsf{sigk_R}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{vk_S}, \mathsf{sigk_S}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and sets $\mathsf{pk_{TS}} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk})$, $\mathsf{sk_R} = (-, \mathsf{sigk_R})$, $\mathsf{pk_S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk_S})$, $\mathsf{sk_S} = (-, \mathsf{sigk_S})$, $\mathsf{pk_{AS}} = \mathsf{PK}$, $\mathsf{sk_{AS}} = \mathsf{DK}$, $\mathsf{pk_{TS}} = \mathsf{PK}'$, and $\mathsf{sk_{TS}} = \mathsf{DK}'$, and sends $(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{sk_{AS}}, \mathsf{pk_{TS}})$ to $\mathcal{A}$. $\mathcal{B}$ sets $\mathsf{SSet} := \emptyset$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to a query kw from $\mathcal{A}$ as follows. $\mathcal{B}$ sends kw to $\mathcal{C}$ as an encryption query. Then, $\mathcal{C}$ generates $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk'_R}, \mathsf{pk'_S}, \mathsf{sk'_S}, kw)$, and sends $\mathsf{ct_{PAEKS}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk_S}, \mathsf{ct_{PAEKS}})$ and $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{AS}}, \sigma||\mathsf{ct_{PAEKS}})$, and sends $\mathsf{ct_{DS\text{-}PAEKS}} = C$ to $\mathcal{A}$.

- For $\mathcal{O}_{\mathrm{Trap}}$, $\mathcal{B}$ can respond to a query kw from $\mathcal{A}$ as follows. $\mathcal{B}$ sets $\mathsf{td}'_{kw} = 0^{|\ell|}$, computes $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk_R}, \mathsf{td}'_{kw})$ and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{TS}}, \sigma'||\mathsf{td}'_{kw})$, and returns $\mathsf{td}_{kw} = C'$ to $\mathcal{A}$. Moreover, $\mathcal{B}$ stores $(kw, 0^{|\ell|}, \sigma')$ to SSet.

- For $\mathcal{O}_{\mathrm{Test}}$, $\mathcal{B}$ can respond to a query $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{ct_{PAEKS}}, \sigma) = 0$. Otherwise, $\mathcal{B}$ runs $\sigma'||\mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk_{TS}}, \mathsf{td}_{kw'})$. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 0$. Otherwise, when $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 1$, σ' has been stored such that $(kw, 0^{|\ell|}, \sigma') \in \mathsf{SSet}$ due to the modification of Game 1. Moreover, $kw \notin \{kw_0^*, kw_1^*\}$ due to the modification of Game 3. Thus, regardless of whether $\mathcal{A}$ has declared (kw_0^*, kw_1^*) or not, $\mathcal{B}$ sends kw to $\mathcal{C}$ as a trapdoor query. $\mathcal{C}$ runs $\mathsf{td}'_{kw} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk'_R}, \mathsf{pk'_S}, \mathsf{sk'_R}, kw)$ and sends td'_{kw} to $\mathcal{B}$. $\mathcal{B}$ returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw})$.

In the challenge phase, $\mathcal{A}$ declares (kw_0^*, kw_1^*). $\mathcal{B}$ sends (kw_0^*, kw_1^*) to $\mathcal{C}$. $\mathcal{C}$ generates the challenge ciphertext $\mathsf{ct^*_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk'_R}, \mathsf{pk'_S}, \mathsf{sk'_S}, kw_\beta^*)$ and sends $\mathsf{ct^*_{PAEKS}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk_S}, \mathsf{ct^*_{PAEKS}})$ and $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{AS}}, \sigma||\mathsf{ct^*_{PAEKS}})$, and sends the challenge ciphertext $\mathsf{ct^*_{DS\text{-}PAEKS}} = C$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs β'. Then $\mathcal{B}$ outputs β'. If $\mathcal{A}$ breaks the IND-AS-CKA security, then $\mathcal{B}$ breaks the IND-CKA security with the same advantage. Thus, $|\Pr[E_3] - 1/2| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{B}}(\lambda)$ holds. □

Now, we have $|\Pr[E_0] - 1/2| \leq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}}(\lambda) + 2q_{\mathsf{Trap}}\mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CKA}}_{\mathsf{PAEKS},\mathcal{B}}(\lambda)$. This concludes the proof of Theorem 1. □

Theorem 2. *The proposed construction is IND-TS-CKA secure if* PAEKS *is IND-CKA secure.*

The adversary $\mathcal{A}$ is allowed to issue a transition query to $\mathcal{O}_{\mathrm{Trans}}$ with no restriction. Thus, $\mathcal{A}$ can obtain $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ for any $\mathsf{ct_{DS\text{-}PAEKS}} = C$. Moreover, $\mathcal{A}$ has the secret key of the test server. That is, $\mathcal{A}$ has $(\mathsf{PK'}, \mathsf{DK'}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and can decrypt $\mathsf{td}_{kw} = C'$ such that $\sigma'||\mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk_{TS}}, \mathsf{td}_{kw'})$. Thus, $\mathcal{A}$ observes PAEKS ciphertexts and trapdoors directly. So, we directly reduce the IND-TS-CKA security to the IND-CKA security.

Proof. Let $\mathcal{A}$ be an adversary of the IND-TS-CKA security and $\mathcal{C}$ be the challenger of the IND-CKA security. We construct an algorithm $\mathcal{B}$ that breaks the IND-CKA security using $\mathcal{A}$ as follows. $\mathcal{C}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk'_R}, \mathsf{sk'_R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp})$, and $(\mathsf{pk'_S}, \mathsf{sk'_S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})$, and sends $(\mathsf{pp}, \mathsf{pk'_R}, \mathsf{pk'_S})$ to $\mathcal{B}$. $\mathcal{B}$ runs $(\mathsf{vk_R}, \mathsf{sigk_R}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{vk_S}, \mathsf{sigk_S}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and $(\mathsf{PK'}, \mathsf{DK'}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and sets $\mathsf{pk_{TS}} = (\mathsf{pk'_R}, \mathsf{vk})$, $\mathsf{sk_R} = (-, \mathsf{sigk_R})$, $\mathsf{pk_S} = (\mathsf{pk'_S}, \mathsf{vk_S})$, $\mathsf{sk_S} = (-, \mathsf{sigk_S})$, $\mathsf{pk_{AS}} = \mathsf{PK}$, $\mathsf{sk_{AS}} = \mathsf{DK}$, $\mathsf{pk_{TS}} = \mathsf{PK'}$, and $\mathsf{sk_{TS}} = \mathsf{DK'}$, and sends $(\mathsf{pp}, \mathsf{pk_R}, \mathsf{pk_S}, \mathsf{pk_{AS}}, \mathsf{pk_{TS}}, \mathsf{sk_{TS}})$ to $\mathcal{A}$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to a query kw from $\mathcal{A}$ as follows. $\mathcal{B}$ sends kw to $\mathcal{C}$ as an encryption query. Then, $\mathcal{C}$ generates $\mathsf{ct_{PAEKS}} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk'_R}, \mathsf{pk'_S}, \mathsf{sk'_S}, kw)$, and sends $\mathsf{ct_{PAEKS}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk_S}, \mathsf{ct_{PAEKS}})$ and $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{AS}}, \sigma||\mathsf{ct_{PAEKS}})$, and sends $\mathsf{ct_{DS\text{-}PAEKS}} = C$ to $\mathcal{A}$.

- For $\mathcal{O}_{\mathrm{Trap}}$, $\mathcal{B}$ can respond to a query kw from $\mathcal{A}$ as follows. Since $kw \notin \{kw_0^*, kw_1^*\}$, $\mathcal{B}$ sends kw to $\mathcal{C}$ as a trapdoor query. $\mathcal{C}$ runs $\mathsf{td}'_{kw} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk'_R}, \mathsf{pk'_S}, \mathsf{sk'_R}, kw)$ and sends

td'_{kw} to $\mathcal{B}$. $\mathcal{B}$ computes $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{R}, \mathsf{td}'_{kw})$ and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{TS}, \sigma' || \mathsf{td}'_{kw})$, and returns $\mathsf{td}_{kw} = C'$ to $\mathcal{A}$.

- For $\mathcal{O}_{\mathrm{Trans}}$, $\mathcal{B}$ can respond to a query $(\mathsf{ct}_{\mathsf{DS\text{-}PAEKS}}, \mathsf{td}_{kw})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ runs $\sigma || \mathsf{ct}_\mathsf{PAEKS} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\mathsf{AS}, C)$. $\mathcal{B}$ returns $\perp$ if $\mathsf{Verify}(\mathsf{vk}_\mathsf{S}, \mathsf{ct}_\mathsf{PAEKS}, \sigma) = 0$. Otherwise, $\mathcal{B}$ returns $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}} = (\mathsf{ct}_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw})$ to $\mathcal{A}$.

In the challenge phase, $\mathcal{A}$ declares (kw_0^*, kw_1^*). $\mathcal{B}$ sends (kw_0^*, kw_1^*) to $\mathcal{C}$. $\mathcal{C}$ generates the challenge ciphertext $\mathsf{ct}^*_\mathsf{PAEKS} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}, kw^*_\beta)$ and sends $\mathsf{ct}^*_\mathsf{PAEKS}$ to $\mathcal{B}$. $\mathcal{B}$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{S}, \mathsf{ct}^*_\mathsf{PAEKS})$ and $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{AS}, \sigma || \mathsf{ct}^*_\mathsf{PAEKS})$, and sends the challenge ciphertext $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}} = C$ to $\mathcal{A}$. We remark that $\mathcal{A}$ may issue $\mathsf{ct}^*_{\mathsf{DS\text{-}PAEKS}}$ to $\mathcal{O}_{\mathrm{Trans}}$ with some td_{kw}. Then, $\mathcal{B}$ simply returns $(\mathsf{ct}^*_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw})$ to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs β'. Then $\mathcal{B}$ outputs β'. If $\mathcal{A}$ breaks the IND-TS-CKA security, then $\mathcal{B}$ breaks the IND-CKA security with the same advantage. This concludes the proof. $\square$

Theorem 3. *The proposed construction is IND-AS-IKGA secure if* PKE *is IND-CCA secure and* Sig *is sEUF-CMA secure.*

Basically, the IND-AS-IKGA security is reduced to the IND-CCA security of PKE because trapdoors are encrypted by the public key of the test server, and the adversary does not have the decryption key. To simulate the test oracle $\mathcal{O}_{\mathrm{Test}}$, PKE is required to be IND-CCA secure because the decryption algorithm of PKE is internally run in the DS-PAEKS.Test algorithm. However, we must consider the following case: how to prevent any modification of DS-PAEKS ciphertexts of the challenge keyword because if an adversary issues $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}}$ to $\mathcal{O}_{\mathrm{Test}}$, where $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}} \leftarrow \mathsf{DS\text{-}PAEKS.Transition}(\mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{AS}, \mathsf{sk}_\mathsf{AS}, \mathsf{ct}_{\mathsf{DS\text{-}PAEKS}}, \mathsf{td}^*_{kw^*_\beta})$, $\mathsf{ct}_{\mathsf{DS\text{-}PAEKS}} \notin \mathsf{CTSet}$, and $\mathsf{ct}_{\mathsf{DS\text{-}PAEKS}}$ is a DS-PAEKS ciphertext of the challenge keyword, then the adversary trivially wins. To handle the issue, we employ the sEUF-CMA security of the underlying signature scheme. That is, a PAEKS ciphertext is signed before encryption that prevents any modification of the DS-PAEKS ciphertext. Then, it is guaranteed that all DS-PAEKS ciphertexts $\mathcal{A}$ obtains are generated by the encryption oracle $\mathcal{O}_C$.

Proof. The proof uses a sequence of games. Let E_i be the event in which $\mathcal{A}$ outputs $\beta' = \beta$ in Game i.

Game 0: This game corresponds to the real game. By definition, $\mathsf{Adv}^{\mathsf{IND\text{-}AS\text{-}IKGA}}_{\mathsf{DS\text{-}PAEKS},\mathcal{A}}(\lambda) = |\Pr[E_0] - 1/2|$.

Game 1: This game is the same as Game 0, except that the response of the $\mathcal{O}_{\mathrm{Test}}$ oracle is changed as follows. Let $\mathcal{A}$ be an IND-AS-CKA adversary. Assume that $\mathcal{A}$ issues $\mathsf{int\text{-}ct}_{\mathsf{DS\text{-}PAEKS}} = (\mathsf{ct}_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw'})$ to $\mathcal{O}_{\mathrm{Test}}$. If $\mathsf{Verify}(\mathsf{vk}_\mathsf{S}, \mathsf{ct}_\mathsf{PAEKS}, \sigma) = 1$ and $(\mathsf{ct}_\mathsf{PAEKS}, \sigma)$ is not generated in the $\mathcal{O}_C$ oracle, then the challenger abort. If the challenger does not abort, then Game 1 is identical to Game 0. Thus, $|\Pr[E_0] - \Pr[E_1]| \leq \Pr[\mathsf{abort}]$ where abort is the event that the challenger aborts.

Lemma 5. *There exists an algorithm $\mathcal{B}$ such that* $\Pr[\mathsf{abort}] \leq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}}(\lambda)$.

Proof. Let $\mathcal{A}$ be the adversary of IND-AS-IKGA and $\mathcal{C}$ be the challenger of the signature scheme. We construct an algorithm $\mathcal{B}$ that breaks the sEUF-CMA security as follows. $\mathcal{B}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{R}, \mathsf{sk}'_\mathsf{R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp})$, $(\mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})$, $(\mathsf{vk}_\mathsf{R}, \mathsf{sigk}_\mathsf{R}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{PK}, \mathsf{DK}) \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$, and $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$. $\mathcal{C}$ runs $(\mathsf{vk}, \mathsf{sigk}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$ and sends vk to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk}_\mathsf{S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk})$, $\mathsf{sk}_\mathsf{R} = (\mathsf{sk}'_\mathsf{R}, \mathsf{sigk}_\mathsf{R})$, $\mathsf{pk}_\mathsf{S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk})$, $\mathsf{sk}_\mathsf{S} =$

$(\mathsf{sk}'_\mathsf{S}, -)$, $\mathsf{pk}_\mathsf{AS} = \mathsf{PK}$, $\mathsf{sk}_\mathsf{AS} = \mathsf{DK}$, $\mathsf{pk}_\mathsf{TS} = \mathsf{PK}'$, and $\mathsf{sk}_\mathsf{TS} = \mathsf{DK}'$, and sends $(\mathsf{pp}, \mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{AS}, \mathsf{sk}_\mathsf{AS}, \mathsf{pk}_\mathsf{TS})$ to $\mathcal{A}$. $\mathcal{B}$ sets $\mathsf{CTSet} := \emptyset$ and $\mathsf{CTSet}' := \emptyset$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to a query kw from $\mathcal{A}$ as follows. $\mathcal{B}$ runs $\mathsf{ct}_\mathsf{PAEKS} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}, kw)$ and sends $\mathsf{ct}_\mathsf{PAEKS}$ to $\mathcal{C}$ as a signing query. $\mathcal{C}$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{S}, \mathsf{ct}_\mathsf{PAEKS})$ and sends σ to $\mathcal{B}$. $\mathcal{B}$ runs $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{AS}, \sigma || \mathsf{ct}_\mathsf{PAEKS})$ and returns $\mathsf{ct}_\mathsf{DS\text{-}PAEKS} = C$ to $\mathcal{A}$. $\mathcal{B}$ stores $(kw, \mathsf{ct}_\mathsf{DS\text{-}PAEKS})$ on CTSet'.
- For $\mathcal{O}_\mathrm{Trap}$, $\mathcal{B}$ can respond to any query from $\mathcal{A}$ because $\mathcal{B}$ knows sk_R.
- For $\mathcal{O}_\mathrm{Test}$, $\mathcal{B}$ can respond to a query $\mathsf{int\text{-}ct}_\mathsf{DS\text{-}PAEKS} = (\mathsf{ct}_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw'})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk}_\mathsf{S}, \mathsf{ct}_\mathsf{PAEKS}, \sigma) = 0$. Otherwise, if $(\mathsf{ct}_\mathsf{PAEKS}, \sigma)$ was not generated in the $\mathcal{O}_C$ oracle, then $(\mathsf{ct}_\mathsf{PAEKS}, \sigma)$ is not a response from $\mathcal{C}$. Thus, $\mathcal{B}$ outputs $(\mathsf{ct}_\mathsf{PAEKS}, \sigma)$ as a forged message and signature pair, and breaks the sEUF-CMA security of the signature scheme. If $(\mathsf{ct}_\mathsf{PAEKS}, \sigma)$ was generated in the $\mathcal{O}_C$ oracle, then $\mathcal{B}$ runs $\sigma' || \mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\mathsf{TS}, \mathsf{td}_{kw'})$. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk}_\mathsf{R}, \mathsf{td}'_{kw'}, \sigma') = 0$. Otherwise, $\mathcal{B}$ returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct}_\mathsf{PAEKS}, \mathsf{td}'_{kw'})$.

In the challenge phase, $\mathcal{A}$ declares (kw_0^*, kw_1^*). $\mathcal{B}$ chooses $\beta \xleftarrow{\$} \{0,1\}$, generates the challenge trapdoor $\mathsf{td}^*_{kw^*_\beta} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}(\mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{AS}, \mathsf{pk}_\mathsf{TS}, kw^*_\beta)$, and sends $\mathsf{td}^*_{kw^*_\beta}$ to $\mathcal{A}$. $\mathcal{B}$ extracts $(kw, \mathsf{ct}_\mathsf{DS\text{-}PAEKS}) \in \mathsf{CTSet}'$ where $kw \in \{kw_0^*, kw_1^*\}$ and adds $\mathsf{ct}_\mathsf{DS\text{-}PAEKS}$ to CTSet.

$\mathcal{B}$ simulates $\mathcal{O}_\mathrm{Trap}$ and $\mathcal{O}_\mathrm{Test}$ oracles as in the previous phase.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to a query kw from $\mathcal{A}$ as follows. $\mathcal{B}$ runs $\mathsf{ct}_\mathsf{PAEKS} \leftarrow \mathsf{PAEKS.Enc}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}, kw)$ and sends $\mathsf{ct}_\mathsf{PAEKS}$ to $\mathcal{C}$ as a signing query. $\mathcal{C}$ runs $\sigma \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{S}, \mathsf{ct}_\mathsf{PAEKS})$ and sends σ to $\mathcal{B}$. $\mathcal{B}$ runs $C \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{AS}, \sigma || \mathsf{ct}_\mathsf{PAEKS})$ and returns $\mathsf{ct}_\mathsf{DS\text{-}PAEKS} = C$ to $\mathcal{A}$. If $kw \in \{kw_0^*, kw_1^*\}$, then $\mathcal{B}$ stores $\mathsf{ct}_\mathsf{DS\text{-}PAEKS}$ on CTSet.

If $\mathcal{A}$ does not issue a test query $\mathsf{int\text{-}ct}_\mathsf{DS\text{-}PAEKS} = (\mathsf{ct}_\mathsf{PAEKS}, \sigma, \mathsf{td}_{kw'})$ where, for $\sigma' || \mathsf{td}'_{kw'} \leftarrow \mathsf{PKE.Dec}(\mathsf{sk}_\mathsf{TS}, \mathsf{td}_{kw'})$, $\mathsf{Verify}(\mathsf{vk}_\mathsf{R}, \mathsf{td}'_{kw'}, \sigma') = 1$ and $(\mathsf{td}'_{kw'}, \sigma')$ was not generated in the $\mathcal{O}_\mathrm{Trap}$ oracle, then $\mathcal{B}$ simulates Game 0, and Game 1 otherwise. $\square$

Game 2: This game is the same as Game 1, except that the challenge trapdoor $\mathsf{td}^*_{kw^*_\beta} \leftarrow \mathsf{DS\text{-}PAEKS.Trapdoor}$ $(\mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{sk}_\mathsf{R}, \mathsf{pk}_\mathsf{AS}, \mathsf{pk}_\mathsf{TS}, kw^*_\beta)$ is generated as follows. Let ℓ be the bit size of PAEKS trapdoor. Set $\mathsf{td}'_{kw^*_\beta} = 0^{|\ell|}$ and run $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{R}, \mathsf{td}'_{kw^*_\beta})$, and $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk}_\mathsf{TS}, \sigma' || \mathsf{td}'_{kw'})$. Set $\mathsf{td}^*_{kw^*_\beta} = C'$.

Lemma 6. *There exists an algorithm $\mathcal{B}$ such that* $|\Pr[E_1] - \Pr[E_2]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$.

Proof. Let $\mathcal{A}$ be the adversary of IND-AS-IKGA and $\mathcal{C}$ be the challenger of the PKE scheme. We construct an algorithm $\mathcal{B}$ that breaks the IND-CCA security as follows. $\mathcal{B}$ runs $\mathsf{pp} \leftarrow \mathsf{PAEKS.Setup}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{R}, \mathsf{sk}'_\mathsf{R}) \leftarrow \mathsf{PAEKS.KG_R}(\mathsf{pp})$, $(\mathsf{vk}_\mathsf{R}, \mathsf{sigk}_\mathsf{R}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$, $(\mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{S}) \leftarrow \mathsf{PAEKS.KG_S}(\mathsf{pp})$, and $(\mathsf{vk}_\mathsf{S}, \mathsf{sigk}_\mathsf{S}) \leftarrow \mathsf{Sig.KeyGen}(1^\lambda)$. $\mathcal{C}$ runs $(\mathsf{PK}', \mathsf{DK}') \leftarrow \mathsf{PKE.KeyGen}(1^\lambda)$ and sends PK' to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{pk}_\mathsf{TS} = (\mathsf{pk}'_\mathsf{R}, \mathsf{vk})$, $\mathsf{sk}_\mathsf{R} = (\mathsf{sk}'_\mathsf{R}, \mathsf{sigk}_\mathsf{R})$, $\mathsf{pk}_\mathsf{S} = (\mathsf{pk}'_\mathsf{S}, \mathsf{vk}_\mathsf{S})$, $\mathsf{sk}_\mathsf{S} = (\mathsf{sk}'_\mathsf{S}, \mathsf{sigk}_\mathsf{S})$, $\mathsf{pk}_\mathsf{AS} = \mathsf{PK}$, $\mathsf{sk}_\mathsf{AS} = \mathsf{DK}$, $\mathsf{pk}_\mathsf{TS} = \mathsf{PK}'$, and $\mathsf{sk}_\mathsf{TS} = -$, and sends $(\mathsf{pp}, \mathsf{pk}_\mathsf{R}, \mathsf{pk}_\mathsf{S}, \mathsf{pk}_\mathsf{AS}, \mathsf{sk}_\mathsf{AS}, \mathsf{pk}_\mathsf{TS})$ to $\mathcal{A}$.

When $\mathcal{A}$ declares (kw_0^*, kw_1^*), then $\mathcal{B}$ chooses $\beta \xleftarrow{\$} \{0,1\}$, computes $\mathsf{td}'_{kw^*_\beta} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw^*_\beta)$, $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{R}, \mathsf{td}'_{kw^*_\beta})$, and $\sigma'' \leftarrow \mathsf{Sign}(\mathsf{sigk}_\mathsf{R}, 0^{|\ell|})$. $\mathcal{B}$ sets $(\sigma' || \mathsf{td}'_{kw^*_\beta}, \sigma'' || 0^{|\ell|})$ as the challenge plaintexts, and sends $(\sigma' || \mathsf{td}'_{kw^*_\beta}, \sigma'' || 0^{|\ell|})$ to $\mathcal{C}$. $\mathcal{C}$ returns the challenge ciphertext C^*. $\mathcal{B}$ sets $\mathsf{td}^*_{kw^*_\beta} = C^*$.

- For $\mathcal{O}_C$, $\mathcal{B}$ can respond to any query because $\mathcal{B}$ has $\mathsf{sk_S}$.
- For $\mathcal{O}_{\text{Trap}}$, $\mathcal{B}$ can respond to any query from $\mathcal{A}$ because $\mathcal{B}$ knows $\mathsf{sk_R}$.
- For $\mathcal{O}_{\text{Test}}$, $\mathcal{B}$ can respond to a query $\mathsf{int\text{-}ct_{DS\text{-}PAEKS}} = (\mathsf{ct_{PAEKS}}, \sigma, \mathsf{td}_{kw'})$ from $\mathcal{A}$ as follows. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_S}, \mathsf{ct_{PAEKS}}, \sigma) = 0$. Now, $(\mathsf{ct_{PAEKS}}, \sigma)$ was generated in the $\mathcal{O}_C$ oracle due to the modification of Game 1. Thus, $\mathcal{B}$ knows the corresponding keyword kw that was sent to $\mathcal{O}_C$ and $\mathcal{O}_C$ returned $(\mathsf{ct_{PAEKS}}, \sigma)$. If $\mathsf{td}_{kw'} = C^*$, then $\mathcal{B}$ knows kw' is either kw_0^* or kw_1^* because $\mathcal{B}$ chooses b. If $kw = kw_\beta^*$, then $\mathcal{B}$ returns 1, and 0 otherwise. If $\mathsf{td}_{kw'} \neq C^*$, then $\mathcal{B}$ sends $\mathsf{td}_{kw'}$ to $\mathcal{C}$ as a decryption query. If $\mathcal{C}$ returns $\perp$, then $\mathcal{B}$ returns 0 to $\mathcal{A}$. Otherwise, let $\sigma'||\mathsf{td}'_{kw'}$ be the response from $\mathcal{C}$. $\mathcal{B}$ returns 0 if $\mathsf{Verify}(\mathsf{vk_R}, \mathsf{td}'_{kw'}, \sigma') = 0$. Otherwise, $\mathcal{B}$ runs $\mathsf{td}'_{kw} \leftarrow \mathsf{PAEKS.Trapdoor}(\mathsf{pk}'_\mathsf{R}, \mathsf{pk}'_\mathsf{S}, \mathsf{sk}'_\mathsf{R}, kw)$, and returns the result of $\mathsf{PAEKS.Test}(\mathsf{ct_{PAEKS}}, \mathsf{td}'_{kw})$.

If the challenge ciphertext C^* is an encryption of $\sigma'||\mathsf{td}'_{kw_\beta^*}$, then $\mathcal{B}$ simulates Game 1, and if C^* is an encryption of $\sigma''||0^{|\ell|}$, then $\mathcal{B}$ simulates Game 2. Thus, $|\Pr[E_1] - \Pr[E_2]| \leq \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$ holds. □

Now $\Pr[E_2] = 0$ because $\mathsf{td}^*_{kw_\beta^*}$ is independent of β and information about b is completely hidden. Thus, we have $|\Pr[E_0] - 1/2| \leq \mathsf{Adv}^{\mathsf{sEUF\text{-}CMA}}_{\mathsf{Sig},\mathcal{B}}(\lambda) + \mathsf{Adv}^{\mathsf{IND\text{-}CCA}}_{\mathsf{PKE},\mathcal{B}}(\lambda)$. This concludes the proof of Theorem 3. □

Theorem 4. *The proposed construction is IND-TS-IKGA secure if* PAEKS *is IND-IKGA secure.*

Proof Sketch. Since $\mathcal{A}$ has the secret key of the test server, $\mathcal{A}$ can decrypt $C' \leftarrow \mathsf{PKE.Enc}(\mathsf{pk_{TS}}, \sigma'||\mathsf{td}'_{kw'})$ and can observe a PAEKS trapdoor $\mathsf{td}'_{kw'}$ directly. Thus, as in IND-TS-CKA, we directly reduce the IND-TS-IKGA security to the IND-IKGA security. The proof is almost the same as that of IND-TS-CKA, and we omit the proof. □

7 Conclusion

In this paper, we propose a generic construction of DS-PAEKS derived from PAEKS, two PKE schemes, and two signature schemes. We also show that the DS-PAEKS scheme [4], the DS-PEKS scheme [5], and the DS-PEKS construction [22] are vulnerable.

Our consistency definition considers the case that a keyword for encryption and a keyword for trapdoor are different. However, a stronger definition has been considered in [12]. It considers a multi-sender setting, where a trapdoor associated with a sender does not work against ciphertexts generated by the secret key of another sender, even if the same keyword is associated. Considering the stronger definition in the DS-PAEKS context is left as a future work of this paper.

Acknowledgment: The main part of study was done when the author was with the National Institute of Information and Communications Technology (NICT), Japan. This work was supported by JSPS KAKENHI Grant Number JP21K11897.

References

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

[2] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[3] Marco Calderini, Riccardo Longo, Massimiliano Sala, and Irene Villa. Searchable encryption with randomized ciphertext and randomized keyword search. *IACR Cryptol. ePrint Arch.*, page 945, 2022.

[4] Biwen Chen, Libing Wu, Sherali Zeadally, and Debiao He. Dual-server public-key authenticated encryption with keyword search. *IEEE Transactions on Cloud Computing*, 10(1):322–333, 2022.

[5] Rongmao Chen, Yi Mu, Guomin Yang, Fuchun Guo, and Xiaofen Wang. Dual-server public-key encryption with keyword search for secure cloud storage. *IEEE Transactions on Information Forensics and Security*, 11(4):789–798, 2016.

[6] 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.

[7] 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.

[8] Leixiao Cheng and Fei Meng. Public key authenticated encryption with keyword search from LWE. In *ESORICS*, pages 303–324, 2022.

[9] Leixiao Cheng and Fei Meng. Server-aided public key authenticated searchable encryption with constant ciphertext and constant trapdoor. *IEEE Transactions on Information Forensics and Security*, 19:1388–1400, 2024.

[10] 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.

[11] Haorui Du, Jianhua Chen, Ming Chen, Cong Peng, and Debiao He. A lightweight authenticated searchable encryption without bilinear pairing for cloud computing. *Wireless Communications and Mobile Computing*, pages 2336685:1–2336685:15, 2022.

[12] Keita Emura. Generic construction of public-key authenticated encryption with keyword search revisited: Stronger security and efficient construction. In *ACM APKC*, pages 39–49, 2022.

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

[14] Xueqiao Liu, Kai He, Guomin Yang, Willy Susilo, Joseph Tonien, and Qiong Huang. Broadcast authenticated encryption with keyword search. In *ACISP*, pages 193–213, 2021.

[15] 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. In *ACM ASIACCS*, pages 423–436, 2022.

[16] Yang Lu and Jiguo Li. Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices. *IEEE Transactions on Mobile Computing*, 21(12):4397–4409, 2022.

[17] Long Meng, Liqun Chen, Yangguang Tian, Mark Manulis, and Suhui Liu. FEASE: Fast and expressive asymmetric searchable encryption. In *USENIX Security Symposium*, 2024, to appear. Available at `https://eprint.iacr.org/2024/054`.

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

[19] 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.

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

[21] 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.

[22] Raylin Tso, Kaibin Huang, Yu-Chi Chen, Sk. Md. Mizanur Rahman, and Tsu-Yang Wu. Generic construction of dual-server public key encryption with keyword search on cloud computing. *IEEE Access*, 8:152551–152564, 2020.

[23] Lisha Yao, Jian Weng, Anjia Yang, Xiaojian Liang, Zhenghao Wu, Zike Jiang, and Lin Hou. Scalable CCA-secure public-key authenticated encryption with keyword search from ideal lattices in cloud computing. *Inf. Sci.*, 624:777–795, 2023.