J Med Syst (2019) 43: 128
https://doi.org/10.1007/s10916-019-1244-2

SYSTEMS-LEVEL QUALITY IMPROVEMENT

A Generic Construction of Integrated Secure-Channel Free PEKS and PKE and its Application to EMRs in Cloud Storage

Tatsuya Suzuki[1,2] · **Keita Emura**[2] · **Toshihiro Ohigashi**[1,2]

Received: 17 September 2018 / Accepted: 24 February 2019 / Published online: 28 March 2019

Abstract

To provide a search functionality for encrypted data, public key encryption with keyword search (PEKS) has been widely recognized. In actual usage, a PEKS scheme should be employed with a PKE scheme since PEKS itself does not support the decryption of data. Since a naive composition of a PEKS ciphertext and a PKE ciphertext does not provide CCA security, several attempts have been made to integrate PEKS and PKE in a joint CCA manner (PEKS/PKE for short). In this paper, we further extend these works by integrating secure-channel free PEKS (SCF-PEKS) and PKE, which we call SCF-PEKS/PKE, where no secure channel is required to send trapdoors. We give a formal security definition of SCF-PEKS/PKE in a joint CCA manner, and propose a generic construction of SCF-PEKS/PKE based on anonymous identity-based encryption, tag-based encryption, and one-time signature. We also strengthen the current consistency definition according to the secure-channel free property, and show that our construction is strongly consistent if the underlying IBE provides unrestricted strong collision-freeness which is defined in this paper. We also show that such an IBE scheme can be constructed by employing the Abdalla et al. transformations (TCC 2010/J. Cryptology 2018). Finally, as an application of SCF-PEKS/PKE, we strengthen the security of encrypted Electronic Medical Record (EMR) system proposed by Guo and Yau (J. Medical Sys. 2015).

Keywords PEKS · Integration of PEKS and PKE · Secure-channel free · Joint CCA security · Encrypted EMR

Introduction

Integration of searchable encryption and public key encryption

Public key encryption with keyword search (PEKS) [7] has been widely recognized as a cryptographic primitive providing a search functionality for encrypted data. Briefly, a trapdoor t_ω is generated with respect to a keyword ω, and one can search a ciphertext of ω by using t_ω. As defined by Abdalla et al. [1], PEKS should provide (wrong keyword) consistency and keyword privacy. Briefly, the former guarantees that for two distinct keywords ω and ω', a ciphertext of ω is not searched by $t_{\omega'}$. The latter guarantees that no information of keyword is revealed from the ciphertext. Abdalla et al. [1] gave a generic construction of PEKS from anonymous identity-based encryption (IBE), e.g., [8, 13, 25].

In actual usage, PEKS should be employed with a PKE scheme since PEKS itself does not support the decryption of data. For example, assume that an e-mail is required to be encrypted. Then, a sender encrypts the mail header or title using a PEKS scheme, and encrypts the mail body using a PKE scheme whose public key is managed by the receiver. Then, a mail gateway can forward the encrypted e-mail by using PEKS, and the receiver can decrypt the ciphertext using their own secret key of the PKE scheme. From now on, we denote the integrated PEKS and PKE as PEKS/PKE as in [36]. As a naive composition, for a PEKS ciphertext C_{PEKS} and a PKE ciphertext C_{PKE}, a ciphertext of PEKS/PKE is described as its concatenation $C_{\mathsf{PEKS}}||C_{\mathsf{PKE}}$.

An extended abstract appears in the 14th International Conference on Information Security Practice and Experience (ISPEC 2018) [33]. This is the full version. In this version, we consider encrypted Electronic Medical Records (EMRs) proposed by Guo and Yau [22] as an application of SCF-PEKS/PKE. We can strengthen the security of the system by integrating SCF-PEKS and PKE. Moreover, we add all security proofs that were omitted in the proceedings version. See Section 2 Differences from the Proceedings Version.

This article is part of the Topical Collection on *Systems-Level Quality Improvement*

✉ Tatsuya Suzuki
t-suzuki@star.tokai-u.jp

Extended author information available on the last page of the article.

Although indistinguishability against chosen ciphertext attack (IND-CCA) is widely recognized as a standard security definition of PKE, obviously, the naive composition does not provide CCA security even if the underlying PKE scheme is CCA secure. For example, the challenge ciphertext $C^*_{\mathsf{PEKS}}||C^*_{\mathsf{PKE}}$ can be modified such as $C_{\mathsf{PEKS}}||C^*_{\mathsf{PKE}}$ where $C_{\mathsf{PEKS}} \neq C^*_{\mathsf{PEKS}}$, and one can send it to the decryption oracle. This was pointed out by Baek et al. [4] who gave a definition of joint CCA security for PEKS/PKE. Later, Zhang and Imai [36] pointed out that Baek et al.'s definition does not consider keyword privacy. They gave a formal definition of PEKS/PKE that captures both data privacy and keyword privacy, and proposed a generic construction of PEKS/PKE. Abdalla et al. [2, 3] further pointed out that there is a room for improvement in the Zhang-Imai model since an adversary is not allowed to access the test oracle in the model. Chen et al. [14] further considered the trapdoor oracle, and proposed a generic construction of PEKS/PKE from (hierarchical) IBE schemes. As concrete constructions, Buccafurri et al. [10] and Saraswat and Sahu [32] proposed PEKS/PKE schemes from (asymmetric) pairings.[1]

Secure-channel free PEKS

In typical usage of PEKS, a receiver generates a trapdoor, and sends it to a server (e.g., mail gateway). Then, since anyone can run the test algorithm when they obtain a trapdoor, the trapdoor must be sent to the server via a secure channel. To remove the secure channel, secure-channel free PEKS (SCF-PEKS), which is also called designated tester PEKS, has been proposed [15–17, 22, 31, 34]. Unlike the case of employing SSL/TLS in a naive way, only the designated server can run the test algorithm even if trapdoors are exposed. In SCF-PEKS, the server also has a public key and a secret key, and a keyword is encrypted by using the server pubic key in addition to the receiver pubic key. The test algorithm is run by using the server secret key in addition to a trapdoor.

Our contribution

As in PEKS, all PEKS/PKE have assumed that trapdoors are sent to the server via a secure channel. In this paper, to remove this limitation we propose PEKS/PKE supporting secure-channel free property, which we call SCF-PEKS/PKE.

First we give a formal security definition of SCF-PEKS/PKE in a joint CCA manner. Basically, we extend the security definition of SCF-PEKS given by Fang et al. [18].[2] We strengthen their consistency definition as follows. First, an adversary is allowed to access the trapdoor oracle in our model. Owing to the secure-channel free property, this setting is natural since trapdoors are sent via a public channel. Moreover, we give the server secret key to the adversary to guarantee that the server has no way of producing inconsistent ciphertexts. We call this weak consistency. We further strengthen the consistency, which we call strong consistency, where (1) an adversary can obtain trapdoors even for challenge keywords, and (2) an adversary is allowed to produce the challenge ciphertext. The first extension is the same as that of unrestricted strong robustness [19], and the second extension is the same as those of strong robustness [2, 3] and strong collision-freeness [29]. For keyword privacy, as in Fang et al., we consider two situations where either an adversary is modeled as the server (then the server secret key is given to the adversary), or an adversary is modeled as a receiver (then the receiver secret key is given to the adversary). In the former, the adversary is allowed to access the trapdoor oracle and the test oracle, and in the latter, the adversary is allowed to access the test oracle. We additionally consider the decryption oracle to integrate SCF-PEKS and PKE in our joint CCA security. We further define data privacy. To guarantee that the server does not obtain information of data via the test procedure, we give the server secret key to the adversary. Moreover, the adversary is allowed to access the decryption oracle.

Second, we propose a generic construction of SCF-PEKS/PKE with weak consistency from anonymous IBE, tag-based encryption (TBE) [27], and a one-time signature (OTS). We also show that our construction is strongly consistent if the underlying anonymous IBE provides unrestricted strong collision-freeness which is implied by unrestricted strong robustness [19]. We will show how to construct these ingredients in Section "Instantiation of our generic construction". Our construction can be seen as an extension of a generic construction of SCF-PEKS from the same ingredients as above, proposed by Emura et al. [16], by considering an observation given by Abdalla et al. [2, 3]. Namely, Abdalla et al. mentioned that if PEKS and PKE support tags, then these can be combined via the Canetti-Halevi-Katz (CHK) transformation [12], leading to a PEKS/PKE scheme secure in the joint CCA manner. That is, by introducing an OTS scheme, a verification key is regarded as a tag of both ciphertexts, and a signature is produced on them. We point out that the Emura et al.

[1] As a similar primitive, decryptable searchable encryption has been proposed [20, 23] where keywords can be recovered from ciphertexts via the decryption procedure. One main difference from PEKS/PKE is that no plaintext space is defined.

[2] Remark that we do not consider security against keyword guessing attacks which is considered by Fang et al. [18], and leave it as a future work of this paper.

construction yields a "tag-based" SCF-PEKS scheme. By introducing a TBE scheme as the underlying PKE scheme supporting tags, we can construct SCF-PEKS/PKE secure in the joint CCA manner. We further modify the construction to protect against re-encryption attacks (See Section "High-level description of our construction" High-level Description of Our Construction for details) by preparing an IBE plaintext to be correlated to a verification key.

Third, as an application of SCF-PEKS/PKE, we employ our SCF-PEKS/PKE to the encrypted Electronic Medical Record (EMR) system proposed by Guo and Yau [22]. Since their scheme is constructed from their SCF-PEKS scheme and a PKE scheme with the naive composition as above, we can strengthen the security of the system in a joint CCA manner. In fact, Guo and Yau mentioned that *we may consider to apply the techniques proposed in* [4, 14, 36] *to combine the public key encryption (PKE) of the EMRs with our SCF-PEKS*, we solve their open problem by proposing SCF-PEKS/PKE.

Differences from the proceedings version

In this version, we give all security proofs which were omitted in the Proceedings version [33] due to the page limitation. See Section "Security analysis" for details. In addition to this, as an application of SCF-PEKS/PKE, we strengthen the security of encrypted EMR system proposed by Guo and Yau [22]. In the Guo-Yau system, a keyword is encrypted by their SCF-PEKS scheme, and *separately*, an electronic medical record is encrypted by a PKE scheme. As mentioned in the introduction section, this naive composition does not provide CCA security even if the underlying PKE scheme is CCA secure. By employing our SCF-PEKS/PKE, we can construct an encrypted EMR system which is secure in the joint CCA manner. See Section "Application to encrypted EMRs in cloud storage" for details.

Preliminaries

In this section, we define the building tools for our generic SCF-PEKS/PKE construction. We denote that $x \xleftarrow{\$} S$ when x is chosen uniformly from a set S. $y \leftarrow A(x)$ means that y is an output of an algorithm A under an input x. We denote *State* as the state information transmitted by the adversary to himself across stages of the attack in experiments.

First, we introduce the definition of TBE [27] as follows. Let $\mathcal{TAG}$ and $\mathcal{M}_{\mathsf{TBE}}$ be a tag space of TBE and a plaintext space of TBE, respectively.

Definition 21 (Syntax of TBE) A TBE scheme TBE consists of the following three algorithms, TBE.KeyGen, TBE.Enc, and TBE.Dec:

- $\mathsf{TBE.KeyGen}(1^\kappa)$: This key generation algorithm takes as an input the security parameter $\kappa \in \mathrm{N}$, and return a public key pk_{TBE} and a secret key sk_{TBE}.
- $\mathsf{TBE.Enc}(pk_{\mathsf{TBE}}, t, M)$: This encryption algorithm takes as input pk_{TBE}, a message $M \in \mathcal{M}_{\mathsf{TBE}}$ with a tag $t \in \mathcal{TAG}$, and returns a ciphertext C_{TBE}.
- $\mathsf{TBE.Dec}(sk_{\mathsf{TBE}}, t, C_{\mathsf{TBE}})$: This decryption algorithm takes as inputs sk_{TBE}, t, and C_{TBE}, and returns a message M or a reject symbol $\perp$.

Correctness is defined as follow: For all $(pk_{\mathsf{TBE}}, sk_{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, all $M \in \mathcal{M}_{TBE}$, and all $t \in \mathcal{TAG}$, $\mathsf{TBE.Dec}(sk_{\mathsf{TBE}}, t, C_{\mathsf{TBE}}) = M$ holds, where $C_{\mathsf{TBE}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathsf{TBE}}, t, M)$.

Next, we define selective-tag weakly secure against chosen ciphertext attack (IND-stag-CCA) as follows.

Definition 22 (IND-stag-CCA) For any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{IND\text{-}stag\text{-}CCA}_{\mathsf{TBE},\mathcal{A}}(\kappa)$ as follows.

$$
\begin{aligned}
&\mathrm{Exp}^{\text{IND-stag-CCA}}_{\mathsf{TBE},\mathcal{A}}(\kappa):\\
&(t^*, State) \leftarrow \mathcal{A}(1^\kappa);\ (pk_{\mathsf{TBE}}, sk_{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)\\
&(M_0^*, M_1^*, State) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{TBE.DEC}}}(\mathsf{find}, pk_{\mathsf{TBE}});\ \mu \xleftarrow{\$} \{0,1\}\\
&C^*_{\mathsf{TBE}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathsf{TBE}}, t^*, M^*_\mu)\\
&\mu' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{TBE.DEC}}}(\mathsf{guess}, C^*_{\mathsf{TBE}}, State)\\
&\text{If } \mu = \mu' \text{ then output 1, and 0 otherwise}
\end{aligned}
$$

- $\mathcal{O}_{\mathsf{TBE.DEC}}$: This decryption oracle takes as input a tag and a ciphertext $(t, C_{\mathsf{TBE}}) \neq (t^*, C^*_{\mathsf{TBE}})$ and returns the result of $\mathsf{TBE.Dec}(sk_{\mathsf{TBE}}, t, C_{\mathsf{TBE}})$.

We say that TBE is IND-stag-CCA secure if the advantage

$$
\begin{aligned}
&\mathrm{Adv}^{\text{IND-stag-CCA}}_{\mathsf{TBE},\mathcal{A}}(\kappa)\\
&:=\left|\ \Pr[\mathrm{Exp}^{\text{IND-stag-CCA}}_{\mathsf{TBE},\mathcal{A}}(\kappa) = 1] - 1/2\ \right|
\end{aligned}
$$

is negligible for any PPT adversary $\mathcal{A}$.

Next, we introduce definition of anonymous IBE with CCA security [21] as follows. Let $\mathcal{ID}$ and $\mathcal{M}_{\mathsf{IBE}}$ be an identity space and a plaintext space of IBE, respectively.

Definition 23 (Syntax of IBE) An IBE scheme IBE consists of the following four algorithms, IBE.Setup, IBE.Extract, IBE.Enc and IBE.Dec:

- IBE.Setup(1^κ): This setup algorithm takes as an input the security parameter $\kappa \in \mathrm{N}$, and return a public key $params$ and a master key mk.
- IBE.Extract($params, mk, ID$): This extract algorithm takes as input an identity $ID \in \mathcal{ID}$ and mk, and returns a secret key sk_{ID} corresponding to ID.
- IBE.Enc($params, ID, M$): This encryption algorithm takes as input $params$, $ID \in \mathcal{ID}$, a message $M \in \mathcal{M}_{\mathsf{IBE}}$, and returns a ciphertext C_{IBE}.
- IBE.Dec($params, sk_{ID}, C_{\mathsf{IBE}}$): This decryption algorithm takes as inputs sk_{ID} and C_{IBE}, and returns a message M or a reject symbol $\perp$.

Correctness is defined as follows: For all $(params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, all $M \in \mathcal{M}_{\mathsf{IBE}}$, and all $ID \in \mathcal{ID}$, $\mathsf{IBE.Dec}(params, sk_{ID}, C_{\mathsf{IBE}}) = M$ holds, where $C_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(params, ID, M)$ and $sk_{ID} \leftarrow \mathsf{IBE.Extract}(params, mk, ID)$.

Next, we define indistinguishability against chosen ciphertext attack (IBE-IND-CCA) as follows.

Definition 24 (IBE-IND-CCA) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{IBE-IND-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa)$ as follows.

$$
\begin{aligned}
&Exp^{\text{IBE-IND-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa):\\
&\quad (params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)\\
&\quad (M_0^*, M_1^*, ID^*, State)\\
&\qquad \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IBE.DEC}}, \mathcal{O}_{\mathsf{IBE.EXTRACT}}}(\mathsf{find}, params)\\
&\quad \mu \xleftarrow{\$} \{0,1\};\ C^*_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(params, ID^*, M^*_\mu)\\
&\quad \mu' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IBE.DEC}}, \mathcal{O}_{\mathsf{IBE.EXTRACT}}}(\mathsf{guess}, C^*_{\mathsf{IBE}}, State)\\
&\quad \text{If } \mu = \mu' \text{ then output 1, and 0 otherwise}
\end{aligned}
$$

- $\mathcal{O}_{\mathsf{IBE.DEC}}$: This decryption oracle takes as input $(ID, C_{\mathsf{IBE}}) \neq (ID^*, C^*_{\mathsf{IBE}})$ and returns the result of $\mathsf{IBE.Dec}(params, sk_{ID}, C_{\mathsf{IBE}})$ where $sk_{ID} \leftarrow \mathsf{IBE.Extract}(params, mk, ID)$.
- $\mathcal{O}_{\mathsf{IBE.EXTRACT}}$: This extract oracle takes as input an identity $ID \neq ID^*$ and returns the corresponding secret key $sk_{ID} \leftarrow \mathsf{IBE.Extract}(params, mk, ID)$.

We say that IBE is IBE-IND-CCA secure if the advantage

$$
\begin{aligned}
&\mathrm{Adv}_{\mathsf{IBE}}, \mathcal{A}^{\text{IBE-IND-CCA}}(\kappa)\\
&:=\mid \Pr[\mathrm{Exp}^{\text{IBE-IND-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa) = 1] - 1/2 \mid
\end{aligned}
$$

is negligible for any PPT adversary.

Next, we define anonymity against chosen-ciphertext attack (IBE-ANO-CCA).

Definition 25 (IBE-ANO-CCA) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{IBE-ANO-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa)$ as follows.

$$
\begin{aligned}
&Exp^{\text{IBE-ANO-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa):\\
&\quad (params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)\\
&\quad (ID_0^*, ID_1^*, M^*, State)\\
&\qquad \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IBE.DEC}}, \mathcal{O}_{\mathsf{IBE.EXTRACT}}}(\mathsf{find}, params)\\
&\quad \mu \xleftarrow{\$} \{0,1\};\ C^*_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(params, ID^*_\mu, M^*)\\
&\quad \mu' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IBE.DEC}}, \mathcal{O}_{\mathsf{IBE.EXTRACT}}}(\mathsf{guess}, C^*_{\mathsf{IBE}}, State)\\
&\quad \text{If } \mu = \mu' \text{ then output 1, and 0 otherwise}
\end{aligned}
$$

- $\mathcal{O}_{\mathsf{IBE.DEC}}$: This decryption oracle takes as input $(ID, C_{\mathsf{IBE}}) \notin \{(ID_0^*, C^*_{\mathsf{IBE}}), (ID_1^*, C^*_{\mathsf{IBE}})\}$ and returns the result of $\mathsf{IBE.Dec}(params, sk_{ID}, C_{\mathsf{IBE}})$ where $sk_{ID} \leftarrow \mathsf{IBE.Extract}(params, mk, ID)$.
- $\mathcal{O}_{\mathsf{IBE.EXTRACT}}$: This extract oracle takes as input $ID \notin \{ID_0^*, ID_1^*\}$ and returns the corresponding secret key $sk_{ID} \leftarrow \mathsf{IBE.Extract}(params, mk, ID)$.

We say that IBE is IBE-ANO-CCA secure if the advantage

$$
\begin{aligned}
&\mathrm{Adv}^{\text{IBE-ANO-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa)\\
&:=\mid \Pr[\mathrm{Exp}^{\text{IBE-ANO-CCA}}_{\mathsf{IBE},\mathcal{A}}(\kappa) = 1] - 1/2 \mid
\end{aligned}
$$

is negligible for any PPT adversary.

Next, we define unrestricted strong collision-freeness where strong means that an adversary is allowed to produce the challenge ciphertext C^*_{IBE}. This is an extension of strong collision-freeness [29]. Informally, strong collision-freeness guarantees that no adversary can produce a ciphertext whose decryption result for two decryption keys are the same, i.e., $M_0^* = M_1^*$. In addition, in our unrestricted strong collision-freeness definition, the trapdoor oracle has no restriction as in unrestricted strong robustness [19]. Informally, unrestricted strong robustness guarantees that no adversary can produce a ciphertext whose decryption result for two decryption keys are both non-$\perp$. Since the condition $M_0^* = M_1^*$ is not required, our unrestricted strong collision-freeness is an intermediate notion where it is weaker than unrestricted strong robustness and is stronger than strong collision-freeness. How to construct an IBE scheme with unrestricted strong collision-freeness is explained in Section "Instantiation of our generic construction".

Definition 26 (Unrestricted Strong Collision-Freeness) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}_{\mathsf{IBE},\mathcal{A}}^{\text{IBE-usCF}}(\kappa)$ as follows.

$Exp_{\mathsf{IBE},\mathcal{A}}^{\text{IBE-usCF}}(\kappa)$:
$(params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$
$(C^*_{\mathsf{IBE}}, ID^*_0, ID^*_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{IBE.EXTRACT}}}(\mathsf{find}, params)$
$sk_{ID^*_0} \leftarrow \mathsf{IBE.Extract}(params, mk, ID^*_0)$
$sk_{ID^*_1} \leftarrow \mathsf{IBE.Extract}(params, mk, ID^*_1)$
$M^*_0 \leftarrow \mathsf{IBE.Dec}(params, sk_{ID^*_0}, C^*_{\mathsf{IBE}})$
$M^*_1 \leftarrow \mathsf{IBE.Dec}(params, sk_{ID^*_1}, C^*_{\mathsf{IBE}})$
If $M^*_0 \neq \perp \wedge M^*_1 \neq \perp \wedge M^*_0 = M^*_1$ then output 1 and 0 otherwise

– $\mathcal{O}_{\mathsf{IBE.EXTRACT}}$: This extract oracle takes as input ID with no restriction, and returns the corresponding secret key $sk_{ID} \leftarrow \mathsf{IBE.Extract}(params, mk, ID)$.

We say that IBE is unrestricted strongly collision-free if the advantage

$$\mathrm{Adv}^{\text{IBE-usCF}}_{\mathsf{IBE},\mathcal{A}}(\kappa) := \Pr[\mathrm{Exp}^{\text{IBE-usCF}}_{\mathsf{IBE},\mathcal{A}}(\kappa) = 1]$$

is negligible for any PPT adversary $\mathcal{A}$.

Next, we introduce OTS [6] as follows. Let $\mathcal{M}_{\mathsf{Sig}}$ be a message space.

Definition 27 (Syntax of OTS) A OTS scheme OTS consists of the following three algorithms, Sig.KeyGen, Sign and Verify:

– $\mathsf{Sig.KeyGen}(1^\kappa)$: This key generation algorithm takes as an input the security parameter $\kappa \in \mathrm{N}$, and returns signing/verification key pair (K_s, K_v).
– $\mathsf{Sign}(K_s, M)$: This signing algorithm takes as inputs K_s and a message $M \in \mathcal{M}_{\mathsf{Sig}}$, and returns a signature σ.
– $\mathsf{Verify}(K_v, M, \sigma)$: This verification algorithm takes as input K_v, M, and σ, and returns 1 (valid) or 0 (invalid).

Correctness is defined as follows: For all $(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$ and all $M \in \mathcal{M}_{\mathsf{Sig}}$, $\mathsf{Verify}(K_v, M, \sigma) = 1$ holds, where $\sigma \leftarrow \mathsf{Sign}(K_s, M)$.

Next, we define strong existential unforgeability against chosen message attack (sEUF-CMA) of OTS as follows.

Definition 28 (one-time sEUF-CMA) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{one-time sEUF-CMA}}_{\mathsf{OTS},\mathcal{A}}(\kappa)$ as follows.

$Exp^{\text{one-time sEUF-CMA}}_{\mathsf{OTS},\mathcal{A}}(\kappa)$:
$(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$; $(M, State) \leftarrow \mathcal{A}(K_v)$
$M \in \mathcal{M}_{\mathsf{Sig}}$; $\sigma \leftarrow \mathsf{Sign}(K_s, M)$
$(M^*, \sigma^*) \leftarrow \mathcal{A}(\sigma, State)$
If $\mathsf{Verify}(K_v, M^*, \sigma^*) = 1$ and $(M^*, \sigma^*) \neq (M, \sigma)$ then output 1, and 0 otherwise

Table 1 Notations

ω	Keyword
M	Plaintext
pk_{S}	Server public key
pk_{R}	Receiver public key
sk_{S}	Server secret key
sk_{R}	Receiver secret key
$\mathcal{K}$	Keyword space
$\mathcal{M}$	Message space
λ	Ciphertext
t_ω	Trapdoor (for ω)

We say that OTS is one-time sEUF-CMA secure if the advantage

$$\mathrm{Adv}^{\text{one-time sEUF-CMA}}_{\mathsf{OTS},\mathcal{A}}(\kappa) := \Pr[\mathrm{Exp}^{\text{one-time sEUF-CMA}}_{\mathsf{OTS},\mathcal{A}}(\kappa) = 1]$$

is negligible for any PPT adversary.

Definitions of SCF-PEKS/PKE

In this section, we define SCF-PEKS/PKE. As in SCF-PEKS, the server and a receiver manage keys separately. A keyword ω and a plaintext M are encrypted by the server public key, pk_{S}, and the receiver public key, pk_{R}. Although a secret key of the receiver, sk_{R}, plays the role of generating trapdoors in SCF-PEKS, we additionally require that sk_{R} plays a role of decrypting a ciphertext. To search for an encrypted keyword, the test algorithm requires both the server secret key, sk_{S}, and the corresponding trapdoor. Let $\mathcal{K}$ be the keyword space and $\mathcal{M}$ be the message space. We sumarize notations in Table 1.

Definition 31 (Syntax of SCF-PEKS/PKE) A SCF-PEKS/PKE scheme SCF-PEKS/PKE consists of the following six algorithms, $\mathsf{SCF\text{-}PEKS/PKE.KeyGen_S}$, $\mathsf{SCF\text{-}PEKS/PKE.KeyGen_R}$, SCF-PEKS/PKE. Trapdoor, SCF-PEKS/PKE.Enc, SCF-PEKS/PKE.Dec and SCF-PEKS/PKE.Test:

– $\mathsf{SCF\text{-}PEKS/PKE.KeyGen_S}(1^\kappa)$: This server key generation algorithm takes as input the security parameter 1^κ ($\kappa \in \mathrm{N}$), and returns a server public key pk_{S} and a server secret key sk_{S}.
– $\mathsf{SCF\text{-}PEKS/PKE.KeyGen_R}(1^\kappa)$: This receiver key generation algorithm takes as input the security parameter 1^κ ($\kappa \in \mathrm{N}$), and returns a receiver public key pk_{R} and a receiver secret key sk_{R}.
– $\mathsf{SCF\text{-}PEKS/PKE.Trapdoor}(pk_{\mathrm{R}}, sk_{\mathrm{R}}, \omega)$: This trapdoor generation algorithm takes as input pk_{R}, sk_{R}, and

a keyword $\omega \in \mathcal{K}$, and returns a trapdoor t_ω corresponding to keyword ω
- SCF-PEKS/PKE.Enc(pk_S, pk_R, ω, M): This encryption algorithm takes as input pk_R, pk_S, ω, and a message $M \in \mathcal{M}$, and returns a ciphertext λ.
- SCF-PEKS/PKE.Dec(pk_R, sk_R, λ): This decryption algorithm takes as input pk_R, sk_R, and λ, and returns a message M or a reject symbol $\perp$.
- SCF-PEKS/PKE.Test(pk_S, sk_S, pk_R, t_ω, λ): This test algorithm takes as input pk_S, sk_S, pk_R, t_ω, and λ, and returns 1 if $\omega = \omega'$, where ω' is the keyword which was used for computing λ, and 0 otherwise.

Correctness is defined as follows: For all $(pk_S, sk_S) \leftarrow$ SCF-PEKS/PKE.KeyGen$_S(1^\kappa)$, all $(pk_R, sk_R) \leftarrow$ SCF-PEKS/PKE.KeyGen$_R(1^\kappa)$, all $\omega \in \mathcal{K}$, and all $M \in \mathcal{M}$, let $\lambda \leftarrow$ SCF-PEKS/PKE.Enc(pk_S, pk_R, ω, M) and $t_\omega \leftarrow$ SCF-PEKS/PKE.Trapdoor(pk_R, sk_R, ω). Then

SCF-PEKS/PKE.Test(pk_S, sk_S, pk_R, t_ω, λ) $= 1$ *and*
SCF-PEKS/PKE.Dec(pk_R, sk_R, λ) $= M$ *holds.*

Next, we define consistency. Basically, consistency guarantees that for two trapdoors t_{ω^*} and $t_{\hat{\omega}^*}$ where $\omega^* \neq \hat{\omega}^*$, a ciphertext of ω^* is not searched by $t_{\hat{\omega}^*}$. We give two definitions. The former case, which we call weak consistency, is essentially the same as that of Chen et al. [14] where the ciphertext λ^* is honestly generated. Due to the secure-channel free setting, we additionally consider the trapdoor oracle, and give sk_S to the adversary.

Definition 32 (Weak Consistency) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}_{\text{SCF-PEKS/PKE},\mathcal{A}}^{\text{WEAK-}textsfCONSIST}(\kappa)$ as follows.

$\mathrm{Exp}_{\text{SCF-PEKS/PKE},\mathcal{A}}^{\text{WEAK-CONSIST}}(\kappa)$:
$(pk_S, sk_S) \leftarrow$ SCF-PEKS/PKE.KeyGen$_S(1^\kappa)$
$(pk_R, sk_R) \leftarrow$ SCF-PEKS/PKE.KeyGen$_R(1^\kappa)$
$(M^*, \omega^*, \hat{\omega}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}}(pk_S, sk_S, pk_R)$
$M^* \in \mathcal{M}$; $\omega^*, \hat{\omega}^* \in \mathcal{K}$; $\omega^* \neq \hat{\omega}^*$
$\lambda^* \leftarrow$ SCF-PEKS/PKE.Enc(pk_S, pk_R, ω^*, M^*)
$t_{\hat{\omega}^*} \leftarrow$ SCF-PEKS/PKE.Trapdoor(pk_R, sk_R, $\hat{\omega}^*$)

If SCF-PEKS/PKE.Test(pk_S, sk_S, pk_R, $t_{\hat{\omega}^*}$, λ^*) $= 1$
then output 1, and 0 otherwise

- $\mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}$: This trapdoor oracle takes as input ω and returns $t_\omega \leftarrow$ SCF-PEKS/PKE.Trapdoor(pk_R, sk_R, ω) where $\omega \notin \{\omega^*, \hat{\omega}^*\}$.

We say that SCF-PEKS/PKE is weakly consistent if the advantage

$$\mathrm{Adv}^{\text{WEAK-CONSIST}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa) := \Pr[\mathrm{Exp}^{\text{WEAK-CONSIST}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa) = 1]$$

is negligible for any PPT adversary $\mathcal{A}$.

Next, we strengthen weak consistency, which we call strong consistency. Here, an adversary is allowed to produce the ciphertext λ^*. This situation is the same as those of strong robustness [2, 3] and strong collision-freeness [29]. Note that, an adversary is not allowed to obtain decryption keys for challenge identities in these models. In our model, the trapdoor oracle has no restriction, i.e., an adversary can obtain trapdoors of challenge keywords. This situation is the same as that of unrestricted strong robustness [19]. Our strong consistency captures the following situation. Owing to the secure-channel free property, an adversary can observe trapdoors. Let the adversary obtain t_{ω^*} and $t_{\hat{\omega}^*}$. Moreover, assume that the adversary knows keywords ω^* and $\hat{\omega}^*$ associated with t_{ω^*} and $t_{\hat{\omega}^*}$, respectively.[3] Then, the adversary may produce a ciphertext where the test algorithm decides that the ciphertext is associated with both ω^* and $\hat{\omega}^*$. Strong consistency prevents this attack.

Definition 33 (Strong Consistency). For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{STRONG-CONSIST}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$ as follows.

$\mathrm{Exp}^{\text{STRONG-CONSIST}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$:
$(pk_S, sk_S) \leftarrow$ SCF-PEKS/PKE.KeyGen$_S(1^\kappa)$
$(pk_R, sk_R) \leftarrow$ SCF-PEKS/PKE.KeyGen$_R(1^\kappa)$
$(\lambda^*, \omega^*, \hat{\omega}^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}}(pk_S, sk_S, pk_R)$
$\omega^*, \hat{\omega}^* \in \mathcal{K}$; $\omega^* \neq \hat{\omega}^*$
$t_{\omega^*} \leftarrow$ SCF-PEKS/PKE.Trapdoor(pk_R, sk_R, ω^*)
$t_{\hat{\omega}^*} \leftarrow$ SCF-PEKS/PKE.Trapdoor(pk_R, sk_R, $\hat{\omega}^*$)
If SCF-PEKS/PKE.Test(pk_S, sk_S, pk_R, t_{ω^*}, λ^*) $= 1$ *and*
SCF-PEKS/PKE.Test(pk_S, sk_S, pk_R, $t_{\hat{\omega}^*}$, λ^*) $= 1$
then output 1, and 0 otherwise

- $\mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}$: This trapdoor oracle takes as input ω with no restriction, and returns $t_\omega \leftarrow$ SCF-PEKS/PKE.Trapdoor(pk_R, sk_R, ω).

We say that SCF-PEKS/PKE is strongly consistent if the advantage

$$\mathrm{Adv}^{\text{STRONG-CONSIST}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa) := \Pr[\mathrm{Exp}^{\text{STRONG-CONSIST}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa) = 1]$$

is negligible for any PPT adversary $\mathcal{A}$.

[3] This assumption is also natural since we do not consider keyword guessing attacks [18].

Next, we define two security notions for keyword privacy, indistinguishability of keywords against chosen keyword attack with the server secret key (IND-CKA-SSK) and indistinguishability of keywords against chosen keyword attack with all trapdoors (IND-CKA-AT). In the IND-CKA-SSK definition, an adversary $\mathcal{A}$ is modeled as the server, and thus sk_S is given to $\mathcal{A}$. If $\mathcal{A}$ obtains trapdoors, then $\mathcal{A}$ can run the test algorithm by myself. Thus, trapdoors of challenge keywords (ω_0^*, ω_1^*) are not given to $\mathcal{A}$. Instead, $\mathcal{A}$ is allowed to access the test oracle for $(\lambda, \omega) \notin \{(\lambda^*, \omega_0^*), (\lambda^*, \omega_1^*)\}$. To guarantee that no information of keyword is revealed via the decryption procedure, $\mathcal{A}$ is allowed to access the decryption oracle with no restriction.

Definition 34 (IND-CKA-SSK). For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{IND-CKA-SSK}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$ as follows. In the experiment, let $\mathcal{O} := \{\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}, \mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}, \mathcal{O}_{\text{SCF-PEKS/PKE.TEST}}\}$.

$\mathrm{Exp}^{\text{IND-CKA-SSK}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$:
$(pk_S, sk_S) \leftarrow \text{SCF-PEKS/PKE.KeyGen}_S(1^\kappa)$
$(pk_R, sk_R) \leftarrow \text{SCF-PEKS/PKE.KeyGen}_R(1^\kappa)$
$(\omega_0^*, \omega_1^*, M^*, State) \leftarrow \mathcal{A}^{\mathcal{O}}(\text{find}, pk_S, sk_S, pk_R)$
$\mu \xleftarrow{\$} \{0, 1\}$
$\lambda^* \leftarrow \text{SCF-PEKS/PKE.Enc}(pk_S, pk_R, \omega_\mu^*, M^*)$
$\mu' \leftarrow \mathcal{A}^{\mathcal{O}}(\text{guess}, \lambda^*, State)$
If $\mu = \mu'$ then output 1, and 0 otherwise

- $\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}$: This decryption oracle takes as input λ with no restriction, and returns the result of $\text{SCF-PEKS/PKE.Dec}(pk_R, sk_R, \lambda)$. Remark that λ^* is also allowed to input.
- $\mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}$: This trapdoor oracle takes as input ω and returns $t_\omega \leftarrow \text{SCF-PEKS/PKE.Trapdoor}(pk_R, sk_R, \omega)$ where $\omega \notin \{\omega_0^*, \omega_1^*\}$.
- $\mathcal{O}_{\text{SCF-PEKS/PKE.TEST}}$: This test oracle takes as input (λ, ω) where $(\lambda, \omega) \notin \{(\lambda^*, \omega_0^*), (\lambda^*, \omega_1^*)\}$, compute $t_\omega \leftarrow \text{SCF-PEKS/PKE.Trapdoor}(pk_R, sk_R, \omega)$, and returns result of $\text{SCF-PEKS/PKE.Test}(pk_S, sk_S, pk_R, t_\omega, \lambda)$.

We say that a SCF-PEKS/PKE scheme SCF-PEKS/PKE is IND-CKA-SSK secure if the advantage

$$\mathrm{Adv}^{\text{IND-CKA-SSK}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa) :=| \Pr[\mathrm{Exp}^{\text{IND-CKA-SSK}(\kappa)=1]-1/2|}_{\text{SCF-PEKS/PKE},\mathcal{A}}$$

is negligible for any PPT adversary $\mathcal{A}$.

Next, we define IND-CKA-AT. In the IND-CKA-AT definition, an adversary $\mathcal{A}$ is modeled as a receiver. Thus, sk_R is given to $\mathcal{A}$. Then, $\mathcal{A}$ can generate trapdoors for all keywords. Since $\mathcal{A}$ does not have sk_S, $\mathcal{A}$ is not allowed to run the test algorithm. Thus, $\mathcal{A}$ is allowed to access the test oracle for $(\lambda, \omega) \notin \{(\lambda^*, \omega_0^*), (\lambda^*, \omega_1^*)\}$. To guarantee that no information of keyword is revealed via the decryption procedure, $\mathcal{A}$ is allowed to access the decryption oracle with no restriction.

Definition 35 (IND-CKA-AT) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{IND-CKA-AT}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$ as follows.

$\mathrm{Exp}^{\text{IND-CKA-AT}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$:
$(pk_S, sk_S) \leftarrow \text{SCF-PEKS/PKE.KeyGen}_S(1^\kappa)$
$(pk_R, sk_R) \leftarrow \text{SCF-PEKS/PKE.KeyGen}_R(1^\kappa)$
$(\omega_0^*, \omega_1^*, M^*, State)$
$\leftarrow \mathcal{A}^{\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}, \mathcal{O}_{\text{SCF-PEKS/PKE.TEST}}}(\text{find}, pk_S, pk_R, sk_R)$
$\mu \xleftarrow{\$} \{0, 1\}$; $\lambda^* \leftarrow \text{SCF-PEKS/PKE.Enc}(pk_S, pk_R, \omega_\mu^*, M^*)$
$\mu' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}, \mathcal{O}_{\text{SCF-PEKS/PKE.TEST}}}(\text{guess}, \lambda^*, State)$
If $\mu = \mu'$ then output 1, and 0 otherwise

- $\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}$: This decryption oracle takes as input λ with no restriction, and returns the result of $\text{SCF-PEKS/PKE.Dec}(pk_R, sk_R, \lambda)$. Remark that λ^* is also allowed to input.
- $\mathcal{O}_{\text{SCF-PEKS/PKE.TEST}}$: This test oracle takes as input $(\lambda, \omega) \notin \{(\lambda^*, \omega_0^*), (\lambda^*, \omega_1^*)\}$, computes $t_\omega \leftarrow \text{SCF-PEKS/PKE.Trapdoor}(pk_R, sk_R, \omega)$, and returns result of $\text{SCF-PEKS/PKE.Test}(pk_S, sk_S, pk_R, t_\omega, \lambda)$.

We say that a SCF-PEKS/PKE scheme SCF-PEKS/PKE is IND-CKA-AT security if the advantage

$$\mathrm{Adv}^{\text{IND-CKA-AT}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa) :=| \Pr[\mathrm{Exp}^{\text{IND-CKA-AT}(\kappa)=1]-1/2|}_{\text{SCF-PEKS/PKE},\mathcal{A}}$$

is negligible for any PPT adversary $\mathcal{A}$.

Next, we define the data privacy for SCF-PEKS/PKE under chosen ciphertext attack with the server secret key and all trapdoors (IND-CCA-SSK/AT) as follows. To guarantee that the server does not obtain any information of plaintext, the adversary $\mathcal{A}$ is given to sk_S. Moreover, to guarantee that no information of plaintext is revealed via the text procedure, $\mathcal{A}$ is allowed to access the trapdoor oracle with no restriction.

Definition 36 (IND-CCA-SSK/AT) For any PPT adversary $\mathcal{A}$ and the security parameter $\kappa \in \mathrm{N}$, we define the experiment $\mathrm{Exp}^{\text{IND-CCA-SSK/AT}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$ as follows.

$\mathrm{Exp}^{\text{IND-CCA-SSK/AT}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)$:
$(pk_S, sk_S) \leftarrow \text{SCF-PEKS/PKE.KeyGen}_S(1^\kappa)$
$(pk_R, sk_R) \leftarrow \text{SCF-PEKS/PKE.KeyGen}_R(1^\kappa)$
$(\omega^*, M_0^*, M_1^*, State)$
$\leftarrow \mathcal{A}^{\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}, \mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}}(\text{find}, pk_S, sk_S, pk_R)$
$\mu \xleftarrow{\$} \{0, 1\}$
$\lambda^* \leftarrow \text{SCF-PEKS/PKE.Enc}(pk_S, pk_R, \omega^*, M_\mu^*)$
$\mu' \leftarrow \mathcal{A}^{\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}, \mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}}(\text{guess}, \lambda^*, State)$
If $\mu = \mu'$ then output 1, and 0 otherwise

– $\mathcal{O}_{\text{SCF-PEKS/PKE.DEC}}$: This decryption oracle takes as input a ciphertext $\lambda \neq \lambda^*$, and returns the result of $\text{SCF-PEKS/PKE.Dec}(pk_{\text{R}}, sk_{\text{R}}, \lambda)$.
– $\mathcal{O}_{\text{SCF-PEKS/PKE.TRAP}}$: This trapdoor oracle takes as input ω with no restriction, and returns $t_\omega \leftarrow \text{SCF-PEKS/PKE.Trapdoor}(pk_{\text{R}}, sk_{\text{R}}, \omega)$. Remark that ω^* is also allowed to input.

We say that a SCF-PEKS/PKE scheme SCF-PEKS/PKE is IND-CCA-SSK/AT secure if the advantage

$$\begin{aligned}&\text{Adv}^{\text{IND-CCA-SSK/AT}}_{\text{SCF-PEKS/PKE},\mathcal{A}}(\kappa)\\ &:=|\ \Pr[\text{Exp}^{\text{IND-CCA-SSK/AT}(\kappa)=1]-1/2|}_{\text{SCF-PEKS/PKE},\mathcal{A}}\end{aligned}$$

is negligible for any PPT adversary $\mathcal{A}$.

Generic construction of SCF-PEKS/PKE

In this section, we propose a generic construction of SCF-PEKS/PKE. We construct SCF-PEKS/PKE from IBE = (IBE,Setup,IBE.Extract, IBE.Enc,IBE.Dec), TBE = (TBE.KeyGen,TBE.Enc,TBE.Dec), and OTS = (Sig.KeyGen, Sign,Verify). Our construction can be seen as an extension of a generic construction of PEKS (from anonymous IBE proposed by Abdalla et al. [1]) and a generic construction of SCF-PEKS (from anonymous IBE, TBE, and OTS proposed by Emura et al. [16]). Thus, we revisit their constructions as follows.

The Abdalla et al. construction

The Abdalla et al. construction is briefly explained as follows. A receiver has the master key mk as its secret key $sk^{\text{IBE}}_{\text{R}}$. A keyword ω is regarded as an identity, i.e., $\mathcal{K}$ is set to $\mathcal{ID}$, and is encrypted as follows. First, a random plaintext $R \in \mathcal{M}_{\text{IBE}}$ is chosen, and next R is encrypted by IBE such that $C_{\text{IBE}} \leftarrow \text{IBE.Enc}(params, \omega, R)$. Then, the PEKS ciphertext is (C_{IBE}, R). A trapdoor t_ω is the decryption key $sk_\omega \leftarrow \text{IBE.Extract}(params, sk^{\text{IBE}}_{\text{R}}, \omega)$. The test algorithm outputs 1 if $\text{IBE.Dec}(params, t_\omega, C_{\text{IBE}}) = R$ holds. Since the underlying IBE is required to be anonymous, no information of ω is revealed from C_{IBE}.

The Emura et al. construction

By additionally employing TBE and OTS, Emura et al. [16] added the secure-channel property to the Abdalla et al. construction. In their construction, the server manages a key pair of TBE $(pk^{\text{TBE}}_{\text{S}}, sk^{\text{TBE}}_{\text{S}})$. A random plaintext $R \in \mathcal{M}_{\text{IBE}}$ is encrypted by IBE, and the IBE ciphertext is encrypted by TBE such that $C_{\text{TBE}} \leftarrow \text{TBE.Enc}(pk^{\text{TBE}}_{\text{S}}, H_{tag}(K_v), C_{\text{IBE}})$, where the verification key K_v is regarded as the tag and $H_{tag} : \{0,1\}^* \rightarrow \mathcal{TAG}$ is a target collision-resistant (TCR) hash function. Finally, a signature is computed such that $\sigma \leftarrow \text{Sign}(K_s, (C_{\text{TBE}}, R))$. The SCF-PEKS ciphertext is $(C_{\text{TBE}}, K_v, \sigma)$. The test algorithm first decrypts C_{TBE} using $sk^{\text{TBE}}_{\text{S}}$, next it decrypts its decryption result using a trapdoor, and then obtains R. The test algorithm outputs 1 if σ is valid on (C_{TBE}, R). Owing to the double encryption, both $sk^{\text{TBE}}_{\text{S}}$ and t_ω are required to run the test algorithm. It is particularly worth noting that the random plaintext R is NOT contained in the ciphertext. Emura et al. mentioned that even if R is contained in a ciphertext, it does not affect the security, and the reason for removing R is to reduce the ciphertext size.

High-level description of our construction

To integrate SCF-PEKS and PKE, the receiver additionally manages a key pair of TBE $(pk^{\text{TBE}}_{\text{R}}, sk^{\text{TBE}}_{\text{R}})$. Since the Emura et al. construction above can be seen as "tag-based" SCF-PEKS, a plaintext $M \in \mathcal{M}_{\text{TBE}}$ is encrypted by $pk^{\text{TBE}}_{\text{R}}$ with the same tag $H_{tag}(K_v)$ such that

$$\begin{aligned}&C_{\text{TBE,S}} \leftarrow \text{TBE.Enc}(pk^{\text{TBE}}_{\text{S}}, H_{tag}(K_v), C_{\text{IBE}})\ and\\ &C_{\text{TBE,R}} \leftarrow \text{TBE.Enc}(pk^{\text{TBE}}_{\text{R}}, H_{tag}(K_v), M)\end{aligned}$$

Here, for the sake of clarity, we use subscript S for ciphertexts encrypted by the server pubic key $pk^{\text{TBE}}_{\text{S}}$, and use subscript R for ciphertexts encrypted by the receiver pubic key $pk^{\text{TBE}}_{\text{R}}$. The sender computes the OTS σ on $(C_{\text{TBE,S}}, C_{\text{TBE,R}}, R)$. A SCF-PEKS/PKE ciphertext is described as $\lambda = (C_{\text{TBE,S}}, C_{\text{TBE,R}}, K_v, \sigma, R)$. It is particularly worth noting that the random plaintext R is contained in the ciphertext unlike in the Emura et al. construction. The ciphertext now provides public verifiability since anyone can verify σ. Since the decryption algorithm needs to verify σ, this public verifiability is necessary.

The construction basically works well since TBE+OTS yields CCA-secure PKE [27]. The main difficulty to be handled is explained as follows. Let $\lambda^* = (C^*_{\text{TBE,S}}, C^*_{\text{TBE,R}}, K^*_v, \sigma^*, R^*)$ be the challenge ciphertext in the IND-CKA-SSK game. Now we consider how to reduce the IND-CKA-SSK security to the IBE-ANO-CCA security. Since the adversary $\mathcal{A}$ has $sk^{\text{TBE}}_{\text{S}}$, $\mathcal{A}$ can decrypt $C^*_{\text{TBE,S}}$. Let C^*_{IBE} be the decryption result. Then, $\mathcal{A}$ can compute a valid ciphertext $\lambda \neq \lambda^*$ such that (1) (K_s, K_v) is chosen by $\mathcal{A}$ with the condition $K_v \neq K^*_v$, (2) C^*_{IBE} is re-encrypted with the tag $H_{tag}(K_v)$ such that $C_{\text{TBE,S}} \leftarrow \text{TBE.Enc}(pk^{\text{TBE}}_{\text{S}}, H_{tag}(K_v), C^*_{\text{IBE}})$, (3) $C_{\text{TBE,R}} \leftarrow \text{TBE.Enc}(pk^{\text{TBE}}_{\text{R}}, H_{tag}(K_v), M)$ is computed with arbitrary M, (4) $\sigma \leftarrow \text{Sign}(K_s, (C_{\text{TBE,S}}, C_{\text{TBE,R}}, R^*))$ is computed, and (5) $\lambda = (C_{\text{TBE,S}}, C_{\text{TBE,R}}, K_v, \sigma, R^*)$ is sent to the test oracle with $\omega \in \{\omega^*_0, \omega^*_1\}$. Although the reduction algorithm obtains C^*_{IBE}, the algorithm cannot send the challenge ciphertext C^*_{IBE} with either ω^*_0 or ω^*_1 to the decryption oracle of IBE. Thus, the security proof fails.

To protect against this re-encryption attack, we modify the plaintext of C_{IBE} as

$C_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(params, \omega, R)$ with $R = H_{ibe}(K_v)$

where $H_{ibe} : \{0,1\}^* \rightarrow \mathcal{M}_{\mathsf{IBE}}$ is a TCR hash function, and the test algorithm checks whether or not $R = H_{ibe}(K_v)$. This structure prevents the adversary from employing different K_v and thus, if C^*_{IBE} appears as above, then $K_v = K^*_v$ must hold unless the TCR property is broken. Since this situation contradicts sEUF-CMA security, our simulation works well.

Since R can be computed from K_v, we can now remove R from λ without losing public verifiability, and an SCF-PEKS/PKE ciphertext is described as $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$.

Our generic construction

We give our construction as follows. Assume that $\mathcal{C}_{\mathsf{IBE}} \subseteq \mathcal{M}_{\mathsf{TBE}}$ and $\mathcal{C}_{\mathsf{TBE}} \times \mathcal{C}_{\mathsf{TBE}} \times \mathcal{M}_{\mathsf{IBE}} \subseteq \mathcal{M}_{\mathsf{Sig}}$, where $\mathcal{C}_{\mathsf{IBE}}$ and $\mathcal{M}_{\mathsf{IBE}}$ are a ciphertext space and plaintext space of IBE respectively, $\mathcal{M}_{\mathsf{TBE}}$ is a plaintext space of TBE, and $\mathcal{M}_{\mathsf{Sig}}$ is a message space of OTS.

The Proposed Construction

- $\mathsf{SCF\text{-}PEKS/PKE.KeyGen_S}(1^\kappa)$: Run $(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$. Output $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}$ and $sk_{\mathrm{S}} = sk_{\mathrm{S}}^{\mathsf{TBE}}$.
- $\mathsf{SCF\text{-}PEKS/PKE.KeyGen_R}(1^\kappa)$: Run $(pk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$ and $(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$. Output $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$ and $sk_{\mathrm{R}} = (sk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}})$. We assume that TCR hash functions $H_{tag} : \{0,1\}^* \rightarrow \mathcal{TAG}$ and $H_{ibe} : \{0,1\}^* \rightarrow \mathcal{M}_{\mathsf{IBE}}$ are contained in pk_{R}.
- $\mathsf{SCF\text{-}PEKS/PKE.Trapdoor}(pk_{\mathrm{R}}, sk_{\mathrm{R}}, \omega)$: Parse $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$ and $sk_{\mathrm{R}} = (sk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}})$. Run $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{IBE}}, \omega)$ and output $t_\omega = sk_\omega$.
- $\mathsf{SCF\text{-}PEKS/PKE.Enc}(pk_{\mathrm{S}}, pk_{\mathrm{R}}, \omega, M)$: Parse $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}$ and $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$. Run $(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$ and compute $t = H_{tag}(K_v)$ and $R = H_{ibe}(K_v)$. Run $C_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(pk_{\mathrm{R}}^{\mathsf{IBE}}, \omega, R)$. Compute $C_{\mathsf{TBE,S}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{S}}^{\mathsf{TBE}}, t, C_{\mathsf{IBE}})$, $C_{\mathsf{TBE,R}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{R}}^{\mathsf{TBE}}, t, M)$, and $\sigma \leftarrow \mathsf{Sign}(K_s, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R))$, and output $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$.
- $\mathsf{SCF\text{-}PEKS/PKE.Dec}(pk_{\mathrm{R}}, sk_{\mathrm{R}}, \lambda)$: Parse $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$, $sk_{\mathrm{R}} = (sk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}})$ and $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$. Compute $R = H_{ibe}(K_v)$. If $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R), \sigma) = 0$, then output $\perp$. Otherwise, compute $t = H_{tag}(K_v)$ and output $M \leftarrow \mathsf{TBE.Dec}(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,R}})$.
- $\mathsf{SCF\text{-}PEKS/PKE.Test}(pk_{\mathrm{S}}, sk_{\mathrm{S}}, pk_{\mathrm{R}}, t_\omega, \lambda)$: Parse $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}$, $sk_{\mathrm{S}} = sk_{\mathrm{S}}^{\mathsf{TBE}}$, $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$, and $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$. Compute $t = H_{tag}(K_v)$, and run $C'_{\mathsf{IBE}} \leftarrow \mathsf{TBE.Dec}(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,S}})$ and $R' \leftarrow \mathsf{IBE.Dec}(pk_{\mathrm{R}}^{\mathsf{IBE}}, t_\omega, C'_{\mathsf{IBE}})$. Output 1 if $R' = H_{ibe}(K_v)$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$ hold, and 0 otherwise.

Security analysis

Obviously, correctness holds if TBE, IBE, and OTS are correct. Next, we prove that following theorems hold.

Theorem 51 *SCF-PEKS/PKE is weakly consistent if IBE is IBE-IND-CPA secure.*

Proof Let $\mathcal{A}$ be an adversary who breaks weak consistency, and $\mathcal{C}$ be the challenger of IBE-IND-CPA security. We construct an algorithm $\mathcal{B}$ that breaks IBE-IND-CPA security as follows. First, $\mathcal{C}$ runs $(params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, and gives $params$ to $\mathcal{B}$. $\mathcal{B}$ sets $pk_{\mathrm{R}}^{\mathsf{IBE}} = params$, runs $(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen_S}(1^\kappa)$ and $(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen_R}(1^\kappa)$, and gives $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}$, $sk_{\mathrm{S}} = sk_{\mathrm{S}}^{\mathsf{TBE}}$, and $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$ to $\mathcal{A}$.

For a trapdoor query ω, $\mathcal{B}$ forwards ω to $\mathcal{C}$ as an extraction query. $\mathcal{C}$ computes $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, mk, \omega)$, and sends sk_ω to $\mathcal{B}$. $\mathcal{B}$ sets $t_\omega = sk_\omega$ and sends t_ω to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs $(M^*, \omega^*, \hat{\omega}^*)$. $\mathcal{B}$ chooses (R^*_0, R^*_1),[4] and sends (R^*_0, R^*_1, ω^*) to $\mathcal{C}$. Then, $\mathcal{C}$ computes the challenge ciphertext C^*_{IBE} and sends it to $\mathcal{B}$. Since $\hat{\omega}^*$ has not been queried by $\mathcal{A}$, $\mathcal{B}$ can send $\hat{\omega}^*$ to $\mathcal{C}$, and can obtain $sk_{\hat{\omega}^*}$. Since $\hat{\omega}^*$ is a special keyword that breaks consistency, $sk_{\hat{\omega}^*}$ works for ciphertexts encrypted by ω^*. Thus, $\mathcal{B}$ runs $R \leftarrow \mathsf{IBE.Dec}(pk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\hat{\omega}^*}, C^*_{\mathsf{IBE}})$, outputs 0 if $R = R^*_0$, and 1 otherwise, and breaks IBE-IND-CPA security. □

Theorem 52 *SCF-PEKS/PKE is strongly consistent if IBE is unresticted strong collision-free.*

Proof Let $\mathcal{A}$ be an adversary who breaks strong consistency, and $\mathcal{C}$ be the challenger of unrestricted strong collision-freeness. We construct an algorithm $\mathcal{B}$ that breaks unrestricted strong collision-freeness as follows. First, $\mathcal{C}$ runs $(params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, and gives $params$ to $\mathcal{B}$. $\mathcal{B}$ sets $pk_{\mathrm{R}}^{\mathsf{IBE}} = params$, runs $(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$ and $(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, and gives $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}$, $sk_{\mathrm{S}} = sk_{\mathrm{S}}^{\mathsf{TBE}}$, and $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$ to $\mathcal{A}$.

[4] Remark that, in our construction, the IBE-part is independent to the plaintext M. Thus, $\mathcal{B}$ can choose the challenge plaintexts (R^*_0, R^*_1) regardless of M^*.

For a trapdoor query ω, $\mathcal{B}$ forwards ω to $\mathcal{C}$ as an extraction query. $\mathcal{C}$ computes $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, mk, \omega)$, and sends sk_ω to $\mathcal{B}$. $\mathcal{B}$ sets $t_\omega = sk_\omega$ and sends t_ω to $\mathcal{A}$. We note that $\mathcal{A}$ may query ω^* and $\hat{\omega}^*$ (which are defined in the challenge phase). Since these queries are not prohibited, $\mathcal{B}$ can forward them to $\mathcal{C}$.

Finally, $\mathcal{A}$ outputs $(\lambda^*, \omega^*, \hat{\omega}^*)$ where $\lambda^* = (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, K^*_v, \sigma^*)$. $\mathcal{B}$ computes $C^*_{\mathsf{IBE}} \leftarrow \mathsf{TBE.Dec}(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,S}})$ where $t = H_{tag}(K^*_v)$. Since the test algorithm outputs 1 for λ^* with t_{ω^*} and with $t_{\hat{\omega}^*}$, respectively, λ^* is a valid ciphertext. Moreover, in our construction a plaintext R is chosen from $\mathcal{M}_{\mathsf{IBE}}$ via H_{ibe}, and is not $\perp$. In addition, since the decryption result of C^*_{IBE} using sk_{ω^*} and that of using $sk_{\hat{\omega}^*}$ must be the same,, i.e., $H_{\mathrm{ibe}}(K^*_v)$ and are not $\perp$. Thus, $\mathcal{B}$ outputs $(C^*_{\mathsf{IBE}}, \omega^*, \hat{\omega}^*)$, and breaks unrestricted strong collision-freeness. □

Theorem 53 *SCF-PEKS/PKE is IND-CKA-SSK secure if IBE is IBE-ANO-CCA secure, OTS is one-time sEUF-CMA secure, and H_{ibe} is a TCR hash function.*

Proof Let $\mathcal{A}$ be an adversary who breaks IND-CKA-SSK security, and $\mathcal{C}$ be the challenger of IBE-ANO-CCA security. We construct an algorithm $\mathcal{B}$ that breaks IBE-ANO-CCA as follows. First, $\mathcal{C}$ runs $(params, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, and gives $params$ to $\mathcal{B}$. $\mathcal{B}$ sets $pk_{\mathrm{R}}^{\mathsf{IBE}} = params$, and runs $(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$ and $(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$. $\mathcal{B}$ gives $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}} = sk_{\mathrm{S}}^{\mathsf{TBE}}$, and $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$ to $\mathcal{A}$.

For a trapdoor query ω, $\mathcal{B}$ forwards ω to $\mathcal{C}$ as an extraction query. $\mathcal{C}$ computes $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, mk, \omega)$ and sends sk_ω to $\mathcal{B}$. $\mathcal{B}$ sends $t_\omega = sk_\omega$ to $\mathcal{A}$.

For a decryption query $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ returns $\perp$ if $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, H_{ibe}(K_v)), \sigma) = 0$. Otherwise, $\mathcal{B}$ computes $t = H_{tag}(K_v)$ and returns the result of $\mathsf{TBE.Dec}(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,R}})$ to $\mathcal{A}$.

For a test query (λ, ω) where $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ forwards ω to $\mathcal{C}$ as an extraction query. $\mathcal{C}$ computes $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, mk, \omega)$ and sends sk_ω to $\mathcal{B}$. $\mathcal{B}$ sets $t_\omega = sk_\omega$, and computes $C'_{\mathsf{IBE}} \leftarrow \mathsf{TBE.Dec}(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,S}})$ and $R' \leftarrow \mathsf{IBE.Dec}(pk_{\mathrm{R}}^{\mathsf{IBE}}, t_\omega, C'_{\mathsf{IBE}})$. Output 1 if $R' = H_{ibe}(K_v)$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$ hold, and 0 otherwise.

In the challenge phase, $\mathcal{A}$ sends $(M^*, \omega_0^*, \omega_1^*)$. $\mathcal{B}$ runs $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$, sets $R^* = H_{ibe}(K_v^*)$, and sends $(R^*, \omega_0^*, \omega_1^*)$ to $\mathcal{C}$. $\mathcal{C}$ computes the challenge ciphertext $C^*_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(pk_{\mathrm{R}}^{\mathsf{IBE}}, \omega_\mu^*, R^*)$ where $\mu \in \{0, 1\}$, and sends C^*_{IBE} to $\mathcal{B}$. $\mathcal{B}$ computes $t^* = H_{tag}(K_v^*)$, $C^*_{\mathsf{TBE,S}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{S}}^{\mathsf{TBE}}, t^*, C^*_{\mathsf{IBE}})$, $C^*_{\mathsf{TBE,R}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{R}}^{\mathsf{TBE}}, t^*, M^*)$, and $\sigma^* \leftarrow \mathsf{Sign}(K_s^*, (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, R^*))$, and sends $\lambda^* = (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, K_v^*, \sigma^*)$ to $\mathcal{A}$.

For a trapdoor query $\omega \notin \{\omega_0^*, \omega_1^*\}$, $\mathcal{B}$ forwards ω to $\mathcal{C}$ as an extraction query. $\mathcal{C}$ computes $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, mk, \omega)$ and sends sk_ω to $\mathcal{B}$. $\mathcal{B}$ sends $t_\omega = sk_\omega$ to $\mathcal{A}$.

For a decryption query $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ computes $R = H_{ibe}(K_v)$, and returns $\perp$ if $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R), \sigma) = 0$. Otherwise, $\mathcal{B}$ computes $t = H_{tag}(K_v)$ and returns the result of $\mathsf{TBE.Dec}(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,R}})$ to $\mathcal{A}$. We remark that if $\lambda = \lambda^*$, then $\mathcal{B}$ simply returns M^*.

For a test query $(\lambda, \omega) \notin \{(\lambda^*, \omega_0^*), (\lambda^*, \omega_1^*)\}$, where $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ responds to the query as follows:

- $\omega \notin \{\omega_0^*, \omega_1^*\}$: In this case, $\mathcal{B}$ forwards ω to $\mathcal{C}$ as an extraction query. $\mathcal{C}$ computes $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, mk, \omega)$ and sends sk_ω to $\mathcal{B}$. $\mathcal{B}$ sets $t_\omega = sk_\omega$, and computes $C'_{\mathsf{IBE}} \leftarrow \mathsf{TBE.Dec}(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,S}})$ and $R' \leftarrow \mathsf{IBE.Dec}(pk_{\mathrm{R}}^{\mathsf{IBE}}, t_\omega, C'_{\mathsf{IBE}})$. Output 1 if $R' = H_{ibe}(K_v)$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$ hold, and 0 otherwise.
- $\omega \in \{\omega_0^*, \omega_1^*\}$: Now $\lambda \neq \lambda^*$. Let $C'_{\mathsf{IBE}} \leftarrow \mathsf{TBE.Dec}(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,S}})$ where $t = H_{tag}(K_v)$. We further consider the following two cases:
  - $C'_{\mathsf{IBE}} \neq C^*_{\mathsf{IBE}}$: $\mathcal{B}$ sends $(\omega, C'_{\mathsf{IBE}})$ to $\mathcal{C}$ as a decryption query. $\mathcal{C}$ returns R'. If $R' = \perp$, then $\mathcal{B}$ returns 0 to $\mathcal{A}$. If not, then $\mathcal{B}$ returns 1 if $R' = H_{ibe}(K_v)$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$ hold, and 0 otherwise.
  - $C'_{\mathsf{IBE}} = C^*_{\mathsf{IBE}}$: Now C'_{IBE} is a ciphertext of $H_{ibe}(K_v^*)$. If $K_v \neq K_v^*$ and $H_{ibe}(K_v) = H_{ibe}(K_v^*)$, then it contradicts the TCR property of H_{ibe}.[5] Thus, we assume that $K_v \neq K_v^*$ and $H_{ibe}(K_v) \neq H_{ibe}(K_v^*)$ hold. Then, $\mathcal{B}$ returns 0 since λ is not a valid ciphertext. If $K_v = K_v^*$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, H_{ibe}(K_v^*)), \sigma) = 0$, then, $\mathcal{B}$ returns 0 since λ is not a valid ciphertext. If $K_v = K_v^*$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$, we call the event Forge, and $\mathcal{B}$ outputs a random bit and aborts. Later, we will show that if Forge happens, then we can construct another algorithm that breaks sEUF-CMA security.

If $\mathcal{B}$ does not abort, the simulation is perfect. Finally, $\mathcal{A}$ outputs $\mu' \in \{0, 1\}$. $\mathcal{B}$ outputs μ', and breaks IBE-ANO-CCA security.

[5] Remark that K_v^* can be chosen in the setup phase. Thus, TCR is enough rather than CR.

Here, we show that if Forge happens, then we can construct another algorithm $\mathcal{B}'$ that breaks sEUF-CMA security. Let $\mathcal{C}'$ be the challenger of sEUF-CMA. $\mathcal{C}'$ runs $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$, and sends K_v^* to $\mathcal{B}'$. $\mathcal{B}'$ setups all keys and thus $\mathcal{B}'$ can respond all queries. In the challenge phase, $\mathcal{A}$ sends $(M^*, \omega_0^*, \omega_1^*)$. $\mathcal{B}'$ sets $R^* = H_{ibe}(K_v^*)$ and computes $C^*_{\mathsf{IBE}} \leftarrow \mathsf{IBE.Enc}(pk_{\mathrm{R}}^{\mathsf{IBE}}, \omega_\mu^*, R^*)$ where $\mu \in \{0, 1\}$ is uniformly chosen. $\mathcal{B}$ computes $t^* = H_{tag}(K_v^*)$, $C^*_{\mathsf{TBE,S}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{S}}^{\mathsf{TBE}}, t^*, C^*_{\mathsf{IBE}})$, and $C^*_{\mathsf{TBE,R}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{R}}^{\mathsf{TBE}}, t^*, M^*)$. $\mathcal{B}'$ sends $(C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, R^*)$ to $\mathcal{C}'$ as a signing query, and obtains σ^*. $\mathcal{B}'$ sends $\lambda^* = (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, K_v^*, \sigma^*)$ to $\mathcal{A}$. For a test query (λ, ω), if the event Forge happens, then $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v^*, \sigma)$ holds. Since $(C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, \sigma) \neq (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, \sigma^*)$ and $\mathsf{Verify}(K_v^*, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, H_{ibe}(K_v^*)), \sigma) = 1$, $\mathcal{B}'$ outputs $((C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, H_{ibe}(K_v^*)), \sigma)$ as a forged signature. This concludes the proof. □

Theorem 54 *SCF-PEKS/PKE is IND-CKA-AT secure if TBE is IND-stag-CCA secure, OTS is one-time sEUF-CMA secure, and H_{tag} is a TCR hash function.*

Let $\mathcal{A}$ be an adversary who breaks IND-CKA-AT security, and $\mathcal{C}$ be the challenger of IND-stag-CCA security. We construct an algorithm $\mathcal{B}$ that breaks IND-stag-CCA as follows. First, $\mathcal{B}$ runs $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$, and sends $t^* = H_{tag}(K_v^*)$ to $\mathcal{C}$. $\mathcal{C}$ runs $(pk_{\mathrm{S}}^{\mathsf{TBE}}, sk_{\mathrm{S}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$ and sends $pk_{\mathrm{S}}^{\mathsf{TBE}}$ to $\mathcal{B}$. $\mathcal{B}$ runs $(pk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{IBE}}) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$ and $(pk_{\mathrm{R}}^{\mathsf{TBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}}) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, and gives $pk_{\mathrm{S}} = pk_{\mathrm{S}}^{\mathsf{TBE}}$, $pk_{\mathrm{R}} = (pk_{\mathrm{R}}^{\mathsf{IBE}}, pk_{\mathrm{R}}^{\mathsf{TBE}})$, and $sk_{\mathrm{R}} = (sk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{TBE}})$ to $\mathcal{A}$.

For a decryption query $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ computes $R = H_{ibe}(K_v)$. If $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 0$, then $\mathcal{B}$ returns $\perp$. Otherwise, when $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 1$, $\mathcal{B}$ responds the query as follows.

- $K_v = K_v^*$: We call this event Forge_1. $\mathcal{B}$ outputs a random bit, and aborts. We can show that we can construct another algorithm that breaks sEUF-CMA security when Forge_1 happens. Since the reduction is almost the same as that of the proof of IND-CKA-SSK security, we omit it.
- $K_v \neq K_v^*$: If $H_{tag}(K_v) = H_{tag}(K_v^*)$, then it contradicts the TCR property of H_{tag}. Thus, we assume that $H_{tag}(K_v) \neq H_{tag}(K_v^*)$ and set $t = H_{tag}(K_v)$. $\mathcal{B}$ returns the result of $\mathsf{TBE.Dec}(sk_{\mathrm{R}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,R}})$.

For a test query (λ, ω) where $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ computes $R = H_{ibe}(K_v)$. If $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 0$, then $\mathcal{B}$ returns 0. Otherwise, when $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 1$, $\mathcal{B}$ responds the query as follows.

- $K_v = K_v^*$: We call this event Forge_2. $\mathcal{B}$ outputs a random bit, and aborts. We can show that we can construct another algorithm that breaks sEUF-CMA security when Forge_2 happens. Since the reduction is almost the same as that of the proof of IND-CKA-SSK security, we omit it.
- $K_v \neq K_v^*$: If $H_{tag}(K_v) = H_{tag}(K_v^*)$, then it contradicts the TCR property of H_{tag}. Thus, we assume that $H_{tag}(K_v) \neq H_{tag}(K_v^*)$ and set $t = H_{tag}(K_v)$. $\mathcal{B}$ sends $(t, C_{\mathsf{TBE,S}})$ to $\mathcal{C}$ as a decryption query. Let C'_{IBE} be the return from $\mathcal{C}$. if $C'_{\mathsf{IBE}} = \perp$, then $\mathcal{B}$ returns 0. Otherwise, $\mathcal{B}$ runs $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathrm{R}}^{\mathsf{IBE}}, sk_{\mathrm{R}}^{\mathsf{IBE}}, \omega)$, sets $t_\omega = sk_\omega$, and runs $R' \leftarrow \mathsf{IBE.Dec}(pk_{\mathrm{R}}^{\mathsf{IBE}}, t_\omega, C'_{\mathsf{IBE}})$. $\mathcal{B}$ returns 1 if $R' = H_{ibe}(K_v)$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$ hold, and 0 otherwise.

In the challenge phase, $\mathcal{A}$ sends $(\omega_0^*, \omega_1^*, M^*)$ to $\mathcal{B}$. $\mathcal{B}$ sets $R^* = H_{ibe}(K_v^*)$, computes $C^*_{\mathsf{IBE},0} \leftarrow \mathsf{IBE.Enc}(pk_{\mathrm{R}}^{\mathsf{IBE}}, \omega_0^*, R^*)$ and $C^*_{\mathsf{IBE},1} \leftarrow \mathsf{IBE.Enc}(pk_{\mathrm{R}}^{\mathsf{IBE}}, \omega_0^*, R^*)$, and sends $(C^*_{\mathsf{IBE},0}, C^*_{\mathsf{IBE},1})$ to $\mathcal{C}$ as the challenge plaintexts. $\mathcal{C}$ computes $C^*_{\mathsf{TBE,S}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{S}}^{\mathsf{TBE}}, t^*, C^*_{\mathsf{IBE},\mu})$ where $\mu \in \{0, 1\}$, and returns $C^*_{\mathsf{TBE,S}}$ to $\mathcal{B}$. $\mathcal{B}$ computes $C^*_{\mathsf{TBE,R}} \leftarrow \mathsf{TBE.Enc}(pk_{\mathrm{R}}^{\mathsf{TBE}}, t^*, M^*)$ and $\sigma^* \leftarrow \mathsf{Sign}(K_s^*, (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, R^*))$, and sends $\lambda^* = (C^*_{\mathsf{TBE,S}}, C^*_{\mathsf{TBE,R}}, K_v^*, \sigma^*)$ to $\mathcal{A}$.

For a decryption query $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ computes $R = H_{ibe}(K_v)$. If $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 0$, then $\mathcal{B}$ returns $\perp$. Otherwise, when $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 1$, $\mathcal{B}$ responds the query as follows.

- $K_v = K_v^*$: We call this event Forge_3. $\mathcal{B}$ outputs a random bit, and aborts. We can show that we can construct another algorithm that breaks sEUF-CMA security when Forge_3 happens. Since the reduction is almost the same as that of the proof of IND-CKA-SSK security, we omit it.
- $K_v \neq K_v^*$: If $H_{tag}(K_v) = H_{tag}(K_v^*)$, then it contradicts the TCR property of H_{tag}. Thus, we assume that $H_{tag}(K_v) \neq H_{tag}(K_v^*)$ and set $t = H_{tag}(K_v)$. $\mathcal{B}$ returns the result of $\mathsf{TBE.Dec}(sk_{\mathrm{R}}^{\mathsf{TBE}}, t, C_{\mathsf{TBE,R}})$. We remark that if $\lambda = \lambda^*$, then $\mathcal{B}$ simply returns M^*.

For a test query $(\lambda, \omega) \notin \{(\lambda^*, \omega_0^*), (\lambda^*, \omega_1^*)\}$ where $\lambda = (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, K_v, \sigma)$, $\mathcal{B}$ computes $R = H_{ibe}(K_v)$. If $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 0$, then $\mathcal{B}$ returns 0. Otherwise, when $\mathsf{Verify}(K_v, \sigma, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R)) = 1$, $\mathcal{B}$ responds the query as follows.

- $K_v = K_v^*$: We call this event Forge_4. $\mathcal{B}$ outputs a random bit, and aborts. We can show that we can construct another algorithm that breaks sEUF-CMA security when Forge_4 happens. Since the reduction is almost the same as that of the proof of IND-CKA-SSK security, we omit it.

– $K_v \neq K_v^*$: If $H_{tag}(K_v) = H_{tag}(K_v^*)$, then it contradicts the TCR property of H_{tag}. Thus, we assume that $H_{tag}(K_v) \neq H_{tag}(K_v^*)$ and set $t = H_{tag}(K_v)$. $\mathcal{B}$ sends $(t, C_{\mathsf{TBE,S}})$ to $\mathcal{C}$ as a decryption query. Let C'_{IBE} be the return from $\mathcal{C}$. if $C'_{\mathsf{IBE}} = \perp$, then $\mathcal{B}$ returns 0. Otherwise, $\mathcal{B}$ runs $sk_\omega \leftarrow \mathsf{IBE.Extract}(pk_{\mathsf{R}}^{\mathsf{IBE}}, sk_{\mathsf{R}}^{\mathsf{IBE}}, \omega)$, sets $t_\omega = sk_\omega$, and runs $R' \leftarrow \mathsf{IBE.Dec}(pk_{\mathsf{R}}^{\mathsf{IBE}}, t_\omega, C'_{\mathsf{IBE}})$. $\mathcal{B}$ returns 1 if $R' = H_{ibe}(K_v)$ and $\mathsf{Verify}(K_v, (C_{\mathsf{TBE,S}}, C_{\mathsf{TBE,R}}, R'), \sigma) = 1$ hold, and 0 otherwise.

If $\mathcal{B}$ does not abort, the simulation is perfect. Finally, $\mathcal{A}$ outputs $\mu' \in \{0, 1\}$. $\mathcal{B}$ outputs μ', and breaks IND-stag-CCA security.

Theorem 55 *SCF-PEKS/PKE is IND-CCA-SSK/AT secure if TBE is IND-stag-CCA secure, OTS is one-time sUF-CMA secure, and H_{tag} is a TCR hash function.*

Since our construction provides public verifiability (i.e., the decryption algorithm can verify σ), the proof of IND-CCA-SSK/AT security is essentially the same as the proof of IND-CCA security from TBE and OTS given by Kiltz [27]. That is, first the reduction algorithm decides K_v^* and the challenge tag $t^* = H_{tag}(K_v^*)$, and the algorithm can access the decryption oracle when $t \neq t^*$. If the algorithm needs to decrypt a ciphertext with t^*, then we can construct another algorithm that breaks sEUF-CMA security. We omit the proof.

Instantiation of our generic construction

For TBE, we can simply employ the Kiltz TBE scheme [27], and for the OTS, we can employ any sEUF-CMA secure OTS scheme, e.g., the Wee OTS scheme [35]. We explain how to construct an IBE scheme that matches our requirements, i.e., with unrestricted strong collision-freeness which defined in this paper, and with IBE-ANO-CCA security. To the best of our knowledge, the strongest notion among several robustnesses and collision-freenesses is complete robustness defined by Farshim et al. [19]. They showed that complete robustness implies unrestricted strong robustness. Since unrestricted strong collision-freeness is implied by unrestricted strong robustness, it is enough to construct an IBE scheme with complete robustness for our purpose. Farshim et al. also showed that the transformation from weakly robust IBE (and commitment with the standard hiding and binding properties) to strongly robust IBE, proposed by Abdalla et al. [2, 3], is already powerful enough to construct completely robust IBE.[6] Moreover, Abdalla et al. also proposed a transformation from IBE to weakly robust IBE. Since these transformations preserve the anonymity and CCA security of the underlying IBE scheme, we can construct an IBE-ANO-CCA secure IBE scheme with unrestricted strong collision-freeness by applying the two Abdalla et al. transformations (from normal to weakly robust, and from weakly robust to strongly robust).

We have three candidates as the underlying IBE scheme.[7] One candidate is the Gentry IBE scheme [21] which is IBE-ANO-CCA secure in the standard model. As another standard model construction, we can employ a variant of the Boyen-Waters IBE scheme [9] that uses the CHK transform to achieve IBE-ANO-CCA security. Although Abdalla et al. [2, 3] mentioned that these schemes are not robust, we can add unrestricted strong collision-freeness property to them via the Abdalla et al. transformations. Other candidate is the CCA-version of the Boneh-Franklin IBE scheme [8] which is IBE-ANO-CCA secure in the random oracle model. The scheme is also known to provide strong robustness. However, it is not clear whether the scheme provides unrestricted strong collision-freeness. Thus, we need to properly employ the Abdalla et al. transformation.

Since unrestricted strong collision-freeness is weaker than complete robustness, employing the two Abdalla et al. transformations as above may be somewhat excessive. Thus, directly and simply constructing an IBE-ANO-CCA secure IBE scheme with unrestricted strong collision-freeness is left as an interesting open problem.

Application to encrypted EMRs in cloud storage

Let us consider the case that health care practitioners can access electronic medical records (EMRs) stored in medical cloud storage with mobile devices. Then, as mentioned by Guo and Yau [22], confidentiality of the stored contents is one of the major concerns of patients [26, 28]. To enhance the security of medical cloud storage, Guo and Yau proposed an encrypted EMRs system with keyword

[6] Farshim et al. [19] showed that a transformation proposed by Mohassel [29] is also powerful enough to construct completely robust IBE,al though the transformation requires the random oracle.

[7] Although other anonymous IBE schemes without random oracles based on simple assumptions have been proposed, we cannot employ them. For example, the Chen et al. IBE scheme [13] and the Jutla-Roy IBE scheme [24, 25] are IBE-ANO-CPA secure. Although Jutla and Roy gave a CCA version, the scheme is not anonymous due to its public verifiability where one can check whether or not a ciphertext is valid for an identity.

search. Their system considered two entities: a cloud service provider (CSP) who manages the cloud storage, and health care practitioners who store EMRs in the cloud storage managed by CSP. Then, to ensure the confidentiality of the medical contents, health care practitioners encrypt EMRs. More concretely, the Guo-Yau system employs the Guo-Yau SCF-PEKS scheme with a many-writer/single-reader setting. Namely, there are many health care practitioners S_i (senders) and another health care practitioner R (receiver). Each S_i encrypts EMRs with a PKE scheme under the public key of R. In addition, a keyword ω associated with the EMRs is encrypted with the Guo-Yau SCF-PEKS scheme under the public key available to both R and CSP. Then R can selectively download certain EMRs that are only related to the keyword ω by generating a trapdoor under his/her private key.

This system is exactly the naive composition that we introduced in Section "Integration of searchable encryption and public key encryption". That is, the ciphertext is malleable even if the underlying PKE scheme is CCA secure. More concretely, CPS can exchange the SCF-PEKS part and the PKE part of a ciphertext, and then R receives unexpected EMRs. In addition to this ciphertext malleability, the consistency definition of Guo and Yau is too weak in the sense that the adversary is not allowed to obtain the server secret key. That is, in the Guo-Yau system context, CPS may modify a ciphertext where the test algorithm searches for a ciphertext associated with an incorrect/unexpected keyword. Though we could not find any specific attack, we cannot guarantee that no such attack exists in the meaning of provable security. Our SCF-PEKS/PKE fits the usage of the Guo-Yau system with the same setting, and ensures stronger security in the joint CCA manner.

The Guo-Yau SCF-PEKS scheme is secure in the random oracle model (ROM) [5]. That is, if a hash function is modeled as a random oracle, then all entities are required to send a random oracle query when they compute a hash value. The random oracle returns a random value that is independent to the input. In the security proof, its programmability is quite useful where a simulator can manage a table and can decide hash values. More concretely the simulator can embed some values which are typically related to the underlying complexity assumptions such as the RSA problem, the discrete logarithm problem and so on. As mentioned above, the random oracle model is somewhat artificial, and as widely recognized, its security is not guaranteed when a hash function is replaced to the actual hash function such as SHA256. More precisely, Canetti et al. [11] showed that there is a protocol which is secure in the random oracle model whereas it is insecure with any concrete hash function. Nevertheless, the random oracle model is still widely employed since it yields efficient constructions. However, theoretically, employing random oracles should be avoided as much as possible. Since our generic construction does not require random oracles, we can construct a SCF-PEKS/PKE scheme without random oracle if the underlying cryptographic primitives are secure without random oracle.

Table 2 Comparison

	Model	Specific or Generic	CCA Security	KGA Resistance
[22]	ROM	Specific	No	Yes
Ours	StdM	Generic	Yes	No

One disadvantage of our construction is inefficiency. For example, if we employ the Kiltz TBE scheme [27], the Wee OTS scheme [35], and the Gentry IBE scheme [21] (with a modification using the Pedersen commitment scheme [30] for providing unrestricted strong robustness, see Section "Instantiation of our generic construction"), then we require 18 exponentiations on $\mathbb{G}$ and 4 exponentiations on $\mathbb{G}_T$ for generating a ciphertext, and 2 pairings, 10 exponentiations on $\mathbb{G}$, and 2 exponentiations on $\mathbb{G}_T$ for running the test algorithm, respectively, where $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a bilinear map. On the other hand, the Guo-Yau SCF-PEKS scheme requires 1 pairing and 4 exponentiations on $\mathbb{G}$ for generating a ciphertext, and 3 pairings and 4 exponentiations on $\mathbb{G}$ for running the test algorithm, respectively. Although the encryption cost is additionally required for SCF-PEKS/PKE, the Guo-Yau scheme is more efficient than the above instantiation, due to the random oracle. Another drawback of our construction beside its inefficiency is that we do not consider security against keyword guessing attacks (KGA) though it is considered by the Guo-Yau SCF-PEKS scheme. As mentioned in [16], proposing a generic construction of SCF-PEKS secure against KGA is still open problem. We leave it as a future work of this paper. We briefly compare the Guo-Yau proposal and ours in Table 2. Here StdM stands for standard model which means without random oracles.

Acknowledgements We thank Dr. Yohei Watanabe for helpful discussion.

Funding Information This work was supported in part by the JSPS KAKENHI Grant Number JP16H02808 and the MIC/SCOPE #162108102.

Compliance with Ethical Standards

Conflict of interests We declare that we have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., and Shi, H., Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. *J. Cryptol.* 21(3):350–391, 2008.
2. Abdalla, M., Bellare, M., and Neven, G., Robust encryption. In: TCC, pp. 480–497, 2010.
3. Abdalla, M., Bellare, M., and Neven, G., Robust encryption. *J. Cryptol.* 31(2):307–350, 2018.
4. Baek, J., Safavi-Naini, R., and Susilo, W., On the integration of public key data encryption and public key encryption with keyword search. In: ISC, pp. 217–232, 2006.
5. Bellare, M., and Rogaway, P., Random oracles are practical: A paradigm for designing efficient protocols. In: ACM Conference on Computer and Communications Security, pp. 62–73, 1993.
6. Bellare, M., and Shoup, S., Two-tier signatures, strongly unforgeable signatures, and fiat-shamir without random oracles. In: Public Key Cryptography, pp. 201–216, 2007.
7. Boneh, D., Crescenzo, G. D., Ostrovsky, R., and Persiano, G., Public key encryption with keyword search. In: EUROCRYPT, pp. 506–522, 2004.
8. Boneh, D., and Franklin, M. K., Identity-based encryption from the weil pairing. In: CRYPTO, pp. 213–229, 2001.
9. Boyen, X., and Waters, B., Anonymous hierarchical identity-based encryption (without random oracles). In: CRYPTO, pp. 290–307, 2006.
10. Buccafurri, F., Lax, G., Sahu, R. A., and Saraswat, V., Practical and secure integrated PKE+PEKS with keyword privacy. In: SECRYPT, pp. 448–453, 2015.
11. Canetti, R., Goldreich, O., and Halevi, S., The random oracle methodology, revisited. *J. ACM* 51(4):557–594, 2004.
12. Canetti, R., Halevi, S., and Katz, J., Chosen-ciphertext security from identity-based encryption. In: EUROCRYPT, pp. 207–222, 2004.
13. Chen, J., Lim, H. W., Ling, S., Wang, H., and Wee, H., Shorter IBE and signatures via asymmetric pairings. In: Pairing-Based Cryptography, pp. 122–140, 2012.
14. Chen, Y., Zhang, J., Lin, D., and Zhang, Z., Generic constructions of integrated PKE and PEKS. *Des. Codes Cryptography* 78(2):493–526, 2016.
15. Emura, K., A generic construction of secure-channel free searchable encryption with multiple keywords. In: NSS, pp. 3–18, 2017.
16. Emura, K., Miyaji, A., Rahman, M. S., and Omote, K., Generic constructions of secure-channel free searchable encryption with adaptive security. *Secur. Commun. Netw.* 8(8):1547–1560, 2015.
17. Fang, L., Susilo, W., Ge, C., and Wang, J., A secure channel free public key encryption with keyword search scheme without random oracle. In: CANS, pp. 248–258, 2009.
18. Fang, L., Susilo, W., Ge, C., and Wang, J., Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inf. Sci.* 238:221–241, 2013.
19. Farshim, P., Libert, B., Paterson, K. G., and Quaglia, E. A., Robust encryption, revisited. In: Public-Key Cryptography, pp. 352–368, 2013.
20. Fuhr, T., and Paillier, P., Decryptable searchable encryption. In: ProvSec, pp. 228–236, 2007.
21. Gentry, C., Practical identity-based encryption without random oracles. In: EUROCRYPT, pp. 445–464, 2006.
22. Guo, L., and Yau, W., Efficient secure-channel free public key encryption with keyword search for EMRs in cloud storage. *J. Med. Syst.* 39(2):11, 2015.
23. Hofheinz, D., and Weinreb, E., Searchable encryption with decryption in the standard model. *IACR Cryptology ePrint Archive* 2008:423, 2008.
24. Jutla, C. S., and Roy, A., Shorter quasi-adaptive NIZK proofs for linear subspaces. In: ASIACRYPT, pp. 1–20, 2013.
25. Jutla, C. S., and Roy, A., Shorter quasi-adaptive NIZK proofs for linear subspaces. *J. Cryptology* 30(4):1116–1156, 2017.
26. Kiah, M. L. M., Nabi, M. S., Zaidan, B. B., and Zaidan, A. A., An enhanced security solution for electronic medical records based on AES hybrid technique with SOAP/XML and SHA-1. *J. Med. Syst.* 37(5):9971, 2013.
27. Kiltz, E., Chosen-ciphertext security from tag-based encryption. In: TCC, pp. 581–600, 2006.
28. Lu, C., Wu, Z., Liu, M., Chen, W., and Guo, J., A patient privacy protection scheme for medical information system. *J. Med. Syst.* 37(6):9982, 2013.
29. Mohassel, P., A closer look at anonymity and robustness in encryption schemes. In: ASIACRYPT, pp. 501–518, 2010.
30. Pedersen, T. P., Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO, pp. 129–140, 1991.
31. Rhee, H. S., Park, J. H., and Lee, D. H., Generic construction of designated tester public-key encryption with keyword search. *Inf. Sci.* 205:93–109, 2012.
32. Saraswat, V., and Sahu, R. A., Short integrated PKE+PEKS in standard model. In: SPACE, pp. 226–246, 2017.
33. Suzuki, T., Emura, K., and Ohigashi, T., A generic construction of integrated secure-channel free PEKS and PKE. In: ISPEC, pp. 69–86, 2018.
34. Wang, T., Au, M. H., and Wu, W., An efficient secure channel free searchable encryption scheme with multiple keywords. In: NSS, pp. 251–265, 2016.
35. Wee, H., Public key encryption against related key attacks. In: Public Key Cryptography, pp. 262–279, 2012.
36. Zhang, R., and Imai, H., Combining public key encryption with keyword search and public key encryption. *IEICE Trans.* 92-D(5): 888–896, 2009.

Affiliations

Tatsuya Suzuki[1,2] · Keita Emura[2] · Toshihiro Ohigashi[1,2]

Keita Emura
k-emura@nict.go.jp

Toshihiro Ohigashi
ohigashi@tsc.u-tokai.ac.jp

1 Tokai University, 2-3-23, Takanawa, Minato-ku, Tokyo 108-8619, Japan

2 National Institute of Information and Communications Technology (NICT), 4-2-1, Nukui-kitamachi, Koganei, Tokyo, 184-8795, Japan