Adaptive Secure-Channel Free Public-Key Encryption with Keyword Search Implies Timed Release Encryption

Keita Emura[1], Atsuko Miyaji[2], and Kazumasa Omote[2]

[1] Center for Highly Dependable Embedded Systems Technology
[2] School of Information Science
Japan Advanced Institute of Science and Technology, 1-1, Asahidai, Nomi, Ishikawa, 923-1292, Japan
{k-emura,miyaji,omote}@jaist.ac.jp

Abstract. As well-known results, timed-release encryption (TRE) and public key encryption scheme with keyword search (PEKS) are very close to identity-based encryption (IBE), respectively. It seems natural that there is a close relationship between TRE and PEKS. However, no explicit bridge has been shown between TRE and PEKS so far. In this paper, we show that TRE can be generically constructed by PEKS with extended functionalities, called secure-channel free PEKS (SCF-PEKS) with adaptive security, and discuss the reason why PEKS and (non-adaptive) SCF-PEKS are not suitable for constructing TRE. In addition to this result, we also show that adaptive SCF-PEKS can be generically constructed by anonymous IBE only. That is, for constructing adaptive SCF-PEKS we do not have to require any additional cryptographic primitive compared to the Abdalla et al. PEKS construction (J. Cryptology 2008), even though adaptive SCF-PEKS requires additional functionalities. This result seems also independently interesting.

1 Introduction

Timed-Release Encryption (TRE): Timed-release encryption (TRE) was proposed by May [22], where even a legitimate recipient cannot decrypt a ciphertext before a semi-trusted time server (TS) sends (or broadcasts) a time-release key s_T assigned with a release time T of the encryptor's choice. As a well-known result, (public key based) TRE is very close to identity-based encryption (IBE). More precisely, generic constructions of TRE based on IBE, public key encryption (PKE), and one-time signature (OTS) have been proposed [8, 21, 24]. Since PKE can be constructed by IBE (and OTS) [6], and digital signature can be constructed by the extraction algorithm of IBE [9], we can say that TRE can be generically constructed by IBE. As an intuition, TRE might be close to other cryptographic primitives which are also close to IBE. So, next we introduce public key encryption scheme with keyword search (PEKS) as such a primitive.

Public key Encryption scheme with Keyword Search (PEKS): PEKS was proposed by Boneh et al. [5]. This scheme considers searching keywords

from encrypted data. Briefly, the flow of PEKS is as follows: A receiver makes a trapdoor t_ω for a keyword ω, and uploads it on an e-mail server. A sender makes an encrypted keyword (which is encrypted by using a keyword ω' and the receiver's public key), and sends it to the server. The server outputs 1 if $\omega = \omega'$, by using t_ω, and 0 otherwise. As a way to construct PEKS, Abdalla et al. [1] showed that a generic construction of PEKS based on anonymous IBE is sufficient. Next, we discuss the relationships among IBE, TRE, and PEKS.

The Relationships among IBE, TRE, and PEKS: One may think that TRE can be generically constructed by PEKS, since IBE (with 1-bit plaintext space) can be constructed by PEKS [5], and TRE can be generically constructed by IBE [8, 24]. However, since generic constructions of TRE based on IBE [8, 24] implicitly[3] require multi-bit plaintext space, we cannot conclude that TRE can be generically constructed by PEKS (so we denote it with "?" in Fig 1 later). There are two easy-to-find ways for showing a relationship between TRE and PEKS: (1) for IBE, show that 1-bit plaintext space is enough to make multi-bit plaintext space (as in the PKE case shown by Myers and Shelat [23]), or (2) propose a generic construction of TRE based on IBE with just 1-bit plaintext space. We do not conclude that these are possible or impossible, but try to establish a bridge between TRE and PEKS from another perspective in this paper. To do so, we revisit PEKS with extended functionalities, called secure-channel free PEKS (SCF-PEKS).

Security Conditions of Previous Secure Channel Free PEKS (SCF-PEKS) Schemes and its Theoretic Extension: PEKS schemes ensure that the server (or an outsider) does not learn anything about keywords chosen by the sender *without trapdoor information*. If trapdoors are revealed, then anyone can execute the test procedure. Therefore, trapdoors cannot be sent via public (i.e., insecure) channels. So, in PEKS schemes, a secure channel (such as secure socket layer (SSL) and transport layer security (TLS)) between a receiver and a server is required, and establishing secure channel requires additional setup costs. To solve this problem, secure channel-free PEKS (SCF-PEKS) have been proposed [2, 15, 16, 19], where the server has a public/secret key pair, and the sender makes an encrypted keyword (which is encrypted by using a keyword ω' and both the server's public key and the receiver's public key), and sends it to the server. The server outputs 1 if $\omega = \omega'$ by using the trapdoor t_ω and its own secret key, and 0 otherwise. Even if t_ω is sent via an insecure channel, no entity (except the server) can run the test procedure.

Next, we discuss the security conditions of the previous SCF-PEKS. The security model considered in [2, 15, 16, 19] does not capture the test queries (i.e., "CPA-like" security). As an exception, Rhee et al. have considered test queries [26]. However, this definition is still weak (i.e., "Unquoted CCA-like" security [23]), where an adversary is not allowed to issue the test queries adaptively. By considering the CCA2 security, SCF-PEKS must be secure even if a

[3] That is, a plaintext of IBE has the form $K_v||(M \oplus r)$, where K_v is a verification key of OTS, M is a plaintext of TRE, and r is a random number.

"malicious-but-legitimate" receiver can be admitted to issue test queries *adaptively*. We insist that this adaptive (i.e., "CCA2-like") security is the natural extension of the SCF-PEKS security theoretically[4] , which is called adaptive SCF-PEKS.

Our Contribution: In this paper, we show the relationships of IBE, TRE, and adaptive SCF-PEKS (dashed arrows in Fig 1).

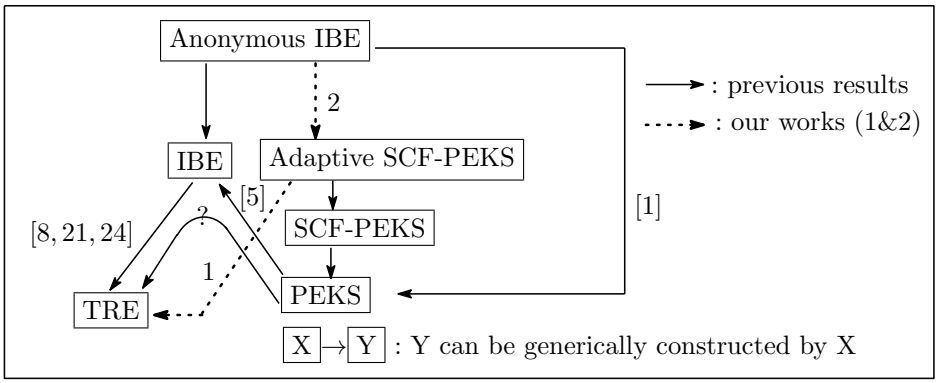

Fig. 1: Relationships of IBE, TRE, and adaptive SCF-PEKS

1. We show that TRE (with 1-bit plaintext space) can be constructed generically from adaptive SCF-PEKS.
   - We discuss the detailed reason why PEKS and (non-adaptive) SCF-PEKS are not suitable for constructing TRE in Section 4.3.
2. We propose a generic construction of adaptive SCF-PEKS based on anonymous IBE, selective-tag chosen-ciphertext (IND-stag-CCA) secure tag-based encryption (TBE), and strongly existentially unforgeable (sUF) OTS. This is the first generic construction of SCF-PEKS.
   - IND-stag-CCA-secure TBE can be constructed by selective-ID chosen plaintext (sID-CPA) secure IBE [20], and digital signature can be constructed by IBE [9]. So, our result shows that adaptive SCF-PEKS can be constructed by anonymous IBE only.
   - No additional cryptographic primitive is required from a generic construction of PEKS [1], even though adaptive SCF-PEKS requires additional functionalities.

2 Preliminaries

In this section, we define the building tools for our generic TRE and adaptive SCF-PEKS construction. $x \xleftarrow{\$} S$ means that x is chosen uniformly from a set

[4] The word "theoretically" means that here we do not discuss the necessity and practicality of adaptive SCF-PEKS. However, since malicious receivers can use the server as the test oracle in the SCF-PEKS usage, our adaptive security notion might be useful in practice.

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.

2.1 Definition of TRE

We refer the Dent et al. TRE definition [10] (which is used by Nakai et al. [24] and Matsuda et al. [21]). As an exception, we exclude pre-open capability from the Dent et al. definition. In the following, $\mathcal{T}$ and $\mathcal{M}_{TRE}$ are a release-time space and a plaintext space, respectively.

TRE scheme Π consists of four algorithms, $\mathsf{TRE.Setup}$, $\mathsf{TRE.UKG}$, $\mathsf{TRE.Ext}$, $\mathsf{TRE.Enc}$, and $\mathsf{TRE.Dec}$. A global parameter prm and a master secret key msk are given by executing $\mathsf{TRE.Setup}(1^\kappa)$. A user's public key pk_u and a user's secret key sk_u are given by executing $\mathsf{TRE.UKG}(1^\kappa)$. For a release-time $T \in \mathcal{T}$, a time-release key s_T corresponding to release-time T is given by executing $\mathsf{TRE.Ext}(\mathsf{prm}, \mathsf{msk}, T)$. For a message $M \in \mathcal{M}_{TRE}$ and $T \in \mathcal{T}$, where $\mathcal{M}_{TRE}$ is the message space of TRE, an encryptor runs $\mathsf{TRE.Enc}(\mathsf{prm}, pk_u, T, M)$, and obtains a ciphertext C_{TRE}. The message M is computed by executing $\mathsf{TRE.Dec}(\mathsf{prm}, sk_u, s_T, C_{TRE})$. Correctness is defined as follows: For all $(\mathsf{prm}, \mathsf{msk}) \leftarrow \mathsf{TRE.Setup}(1^\kappa)$, all $(pk_u, sk_u) \leftarrow \mathsf{TRE.UKG}(1^\kappa)$, all $M \in \mathcal{M}_{TRE}$, and all $T \in \mathcal{T}$, $\mathsf{TRE.Dec}(\mathsf{prm}, sk_u, s_T, C) = M$ holds, where $C \leftarrow \mathsf{TRE.Enc}(\mathsf{prm}, pk_u, T, M)$ and $s_T \leftarrow \mathsf{TRE.Ext}(\mathsf{prm}, \mathsf{msk}, T)$.

Next, we define time-server security, called IND-TR-$\mathrm{CCA}_{\mathrm{TS}}$. It guarantees that no TS can decrypt a ciphertext.

Definition 1 (Time-server Security). *A TRE scheme Π is said to be IND-TR-CCA_{TS} secure if the advantage is negligible for any PPT adversary $\mathcal{A}$, where*

$$
\begin{aligned}
Adv^{\text{IND-TR-CCA}_{TS}}_{\Pi,\mathcal{A}}(1^\kappa) = \big| \Pr \big[& (\mathsf{prm}, \mathsf{msk}) \leftarrow \mathsf{TRE.Setup}(1^\kappa); \\
& (pk_u, sk_u) \leftarrow \mathsf{TRE.UKG}(1^\kappa);\ (M_0^*, M_1^*, T^*, State) \leftarrow \mathcal{A}^{\mathcal{DEC}}(\mathsf{find}, \mathsf{prm}, \mathsf{msk}, pk_u); \\
& \mu \xleftarrow{\$} \{0,1\};\ C^*_{TRE} \leftarrow \mathsf{TRE.Enc}(\mathsf{prm}, pk_u, T^*, M^*_\mu);\ \mu' \leftarrow \mathcal{A}^{\mathcal{DEC}}(\mathsf{guess}, C^*_{TRE}, State); \\
& \mu = \mu' \big] - 1/2 \big|
\end{aligned}
$$

that $\mathcal{DEC}$ is a decryption oracle, where, for input of a ciphertext C_{TRE} and T, it returns the corresponding plaintext M. Note that (C^*_{TRE}, T^*) are not allowed as input to $\mathcal{DEC}$.

Next, we define insider security, called IND-TR-$\mathrm{CPA}_{\mathrm{IS}}$. It guarantees that no receiver can decrypt a ciphertext before the corresponding time-release key is published.

Definition 2 (Insider Security). *A TRE scheme Π is said to be IND-TR-CPA_{IS} secure if the advantage is negligible for any PPT adversary $\mathcal{A}$, where*

$$Adv_{\Pi,\mathcal{A}}^{IND\text{-}TR\text{-}CPA_{IS}}(1^\kappa) =$$
$$\Big|\Pr\big[(\mathsf{prm},\mathsf{msk}) \leftarrow \mathsf{TRE.Setup}(1^\kappa);\ (pk_u, sk_u) \leftarrow \mathsf{TRE.UKG}(1^\kappa);$$
$$(M_0^*, M_1^*, T^*, State) \leftarrow \mathcal{A}^{\mathcal{EXTRACT}}(\mathsf{find}, \mathsf{prm}, pk_u, sk_u);\ \mu \xleftarrow{\$} \{0,1\};$$
$$C_{TRE}^* \leftarrow \mathsf{TRE.Enc}(\mathsf{prm}, pk_u, T^*, M_\mu^*);\ \mu' \leftarrow \mathcal{A}^{\mathcal{EXTRACT}}(\mathsf{guess}, C_{TRE}^*, State);$$
$$\mu = \mu'\big] - 1/2\Big|$$

that $\mathcal{EXTRACT}$ is an extract oracle, where, for input of T, it returns the corresponding time-release key s_T. $T \geq T^*$ is not allowed as input to $\mathcal{EXTRACT}$.

One may think that the notion "CPA" is weak, and a stronger notion can be defined. Actually, the TRE definition [7, 14] achieves such strong security, where $\mathcal{A}$ can access the decryption oracle. However, if no other public key ($pk \neq pk_u$) is considered, such decryption oracle is redundant, since $\mathcal{A}$ has sk_u. Therefore, as in [10, 21, 24], we adopt the CPA notion in this paper.

2.2 Definitions of sUF OTS

A strongly existentially unforgeable (sUF) OTS against adaptively chosen message attack (CMA) [4] Π consists of three algorithms, Sig.KeyGen, Sign and Verify. Sig.KeyGen is a probabilistic algorithm which outputs a signing/verification key pair (K_s, K_v) from the security parameter 1^κ. Sign is a probabilistic algorithm which outputs a signature σ from K_s, and a message $M \in \mathcal{M}_{Sig}$, where $\mathcal{M}_{Sig}$ is the message space of a signature scheme. Verify is a deterministic algorithm which outputs a bit (1 means that σ is a valid signature, and 0 otherwise) from $\sigma \in \mathcal{S}_{sig}$, K_v and M, where $\mathcal{S}_{sig}$ is the signature space. Correctness is defined as follows: For all $(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$ and all $M \in \mathcal{M}_{Sig}$, $\mathsf{Verify}(K_v, \sigma, M) = 1$ holds, where $\sigma \leftarrow \mathsf{Sign}(K_s, M)$.

Definition 3 (one-time sUF-CMA). *A signature scheme is said to be one-time sUF-CMA secure if the advantage $Adv_{\Pi,\mathcal{A}}^{one\text{-}time\ sUF\text{-}CMA}(1^\kappa)$ is negligible for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ in the following experiment.*

$$Adv_{\Pi,\mathcal{A}}^{one\text{-}time\ sUF\text{-}CMA}(1^\kappa) := \Pr\big[(K_s, K_v) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa);$$
$$(M, State) \leftarrow \mathcal{A}(K_v); \sigma \leftarrow \mathsf{Sign}(K_s, M); (M^*, \sigma^*) \leftarrow \mathcal{A}(State, \sigma);$$
$$(M^*, \sigma^*) \neq (M, \sigma); \mathsf{Verify}(K_v, \sigma^*, M^*) = 1\big]$$

2.3 Definitions of IND-stag-CCA Secure TBE

A TBE scheme [20] Π consists of three algorithms, TBE.KeyGen, TBE.Enc and TBE.Dec. The public key pk and the secret key sk are given by executing $\mathsf{TBE.KeyGen}(1^\kappa)$, where $\kappa \in \mathbb{N}$ is the security parameter. For a message $M \in$

$\mathcal{M}_{TBE}$ with a tag $t \in \mathcal{TAG}$, where $\mathcal{M}_{TBE}$ is the message space and $\mathcal{TAG}$ is the tag space of TBE, an encryptor runs $\mathsf{TBE.Enc}(pk, t, M)$, and obtains a ciphertext C_{TBE}. The message M is computed by executing $\mathsf{TBE.Dec}(sk, t, C_{TBE})$. Correctness is defined as follows: For all $(pk, sk) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, all $M \in \mathcal{M}_{TBE}$, and all $t \in \mathcal{TAG}$, $\mathsf{TBE.Dec}(sk, t, C_{TBE}) = M$ holds, where $C_{TBE} \leftarrow \mathsf{TBE.Enc}(pk, t, M)$.

The security experiment of TBE under selective-tag CCA (IND-stag-CCA) is defined as follows.

Definition 4 (IND-stag-CCA). *A TBE scheme is said to be IND-stag-CCA secure if the advantage is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
Adv^{\textit{IND-stag-CCA}}_{\Pi,\mathcal{A}}(1^\kappa) = \big| \Pr \big[& (t^*, State) \leftarrow \mathcal{A}(1^\kappa);\ (pk, sk) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa); \\
& (M_0^*, M_1^*, State) \leftarrow \mathcal{A}^{\mathcal{DEC}}(\mathsf{find}, pk, State);\ \mu \xleftarrow{\$} \{0,1\}; \\
& C^*_{TBE} \leftarrow \mathsf{TBE.Enc}(pk, t^*, M^*_\mu);\ \mu' \leftarrow \mathcal{A}^{\mathcal{DEC}}(\mathsf{guess}, C^*, State); \mu = \mu' \big] - 1/2 \big|
\end{aligned}
$$

that $\mathcal{DEC}$ is a decryption oracle for any tag $t \neq t^*$, where for input of a ciphertext $(C_{TBE}, t) \neq (C^*_{TBE}, t^*)$, it returns the corresponding plaintext M. Note that (C^*_{TBE}, t^*) is not allowed as input to $\mathcal{DEC}$.

2.4 Definitions of Anonymous IBE

IBE scheme Π consists of four algorithms, $\mathsf{IBE.Setup}$, $\mathsf{IBE.Extract}$, $\mathsf{IBE.Enc}$ and $\mathsf{IBE.Dec}$. The public key pk and the master key mk are given by executing $\mathsf{IBE.Setup}(1^\kappa)$. For an identity $ID \in \mathcal{ID}$, where $\mathcal{ID}$ is the identity space, a secret key corresponding to ID sk_{ID} is given by executing $\mathsf{IBE.Extract}(pk, mk, ID)$. For a message $M \in \mathcal{M}_{IBE}$ and $ID \in \mathcal{ID}$, where $\mathcal{M}_{IBE}$ is the message space of IBE, an encryptor runs $\mathsf{IBE.Enc}(pk, ID, M)$, and obtains a ciphertext C_{IBE}. The message M is computed by executing $\mathsf{IBE.Dec}(sk_{ID}, C_{IBE})$. Correctness is defined as follows: For all $(pk, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, all $M \in \mathcal{M}_{IBE}$, and all $ID \in \mathcal{ID}$, $\mathsf{IBE.Dec}(sk_{ID}, C_{IBE}) = M$ holds, where $C_{IBE} \leftarrow \mathsf{IBE.Enc}(pk, ID, M)$ and $sk_{ID} \leftarrow \mathsf{IBE.Extract}(pk, mk, ID)$.

Next, we define the security experiment of IBE under chosen ciphertext attack (IBE-IND-CCA) as follows.

Definition 5 (IBE-IND-CCA). *An IBE scheme is said to be IBE-IND-CCA secure if the advantage is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
Adv^{\textit{IBE-IND-CCA}}_{\Pi,\mathcal{A}}(1^\kappa) = \big| \Pr \big[& (pk, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa); \\
& (M_0^*, M_1^*, ID^*, State) \leftarrow \mathcal{A}^{\mathcal{EXTRACT}, \mathcal{DEC}}(\mathsf{find}, pk);\ \mu \xleftarrow{\$} \{0,1\}; \\
& C^*_{IBE} \leftarrow \mathsf{IBE.Enc}(pk, ID^*, M^*_\mu);\ \mu' \leftarrow \mathcal{A}^{\mathcal{EXTRACT}, \mathcal{DEC}}(\mathsf{guess}, C^*_{IBE}, State); \\
& \mu = \mu' \big] - 1/2 \big|
\end{aligned}
$$

that $\mathcal{EXTRACT}$ is an extract oracle, where, for input of an identity ID, it returns the corresponding secret key sk_{ID}. ID^* is not allowed as input to $\mathcal{EXTRACT}$. $\mathcal{DEC}$ is a decryption oracle, where, for input of a ciphertext C and an identity ID, it returns the corresponding plaintext M. (ID^*, C^*_{IBE}) is not allowed as input to $\mathcal{DEC}$. Chosen plaintext security (IBE-IND-CPA) is simply defined by removing $\mathcal{DEC}$ from the IBE-IND-CCA experiment.

Next, we define anonymity experiment of IBE under CPA (IBE-ANO-CPA).

Definition 6 (IBE-ANO-CPA). *An IBE scheme is said to be IBE-ANO-CPA secure if the advantage is negligible for any PPT adversary $\mathcal{A}$, where*

$$
\begin{aligned}
Adv^{IBE\text{-}ANO\text{-}CPA}_{\Pi,\mathcal{A}}(1^\kappa) = \big|\Pr\big[&(pk, mk) \leftarrow \mathsf{IBE.Setup}(1^\kappa);\\
&(ID^*_0, ID^*_1, M^*, State) \leftarrow \mathcal{A}^{\mathcal{EXTRACT}}(\mathsf{find}, pk);\ \mu \xleftarrow{\$} \{0,1\};\\
&C^*_{IBE} \leftarrow \mathsf{IBE.Enc}(pk, ID^*_\mu, M^*);\ \mu' \leftarrow \mathcal{A}^{\mathcal{EXTRACT}}(\mathsf{guess}, C^*_{IBE}, State);\\
&\mu = \mu'\big] - 1/2\big|
\end{aligned}
$$

Note that ID^*_0 and ID^*_1 are not allowed as input to $\mathcal{EXTRACT}$.

Definition 7 (Anonymous IBE). *An IBE scheme is said to be anonymous IBE if the IBE scheme is both IBE-IND-CPA secure and IBE-ANO-CPA secure.*

3 Definitions of Adaptive SCF-PEKS

In this section, we define security requirements of SCF-PEKS. An SCF-PEKS scheme Π consists of five algorithms, $\mathsf{SCF\text{-}PEKS.KeyGen_S}$, $\mathsf{SCF\text{-}PEKS.KeyGen_R}$, $\mathsf{SCF\text{-}PEKS.Trapdoor}$, $\mathsf{SCF\text{-}PEKS.Enc}$ and $\mathsf{SCF\text{-}PEKS.Test}$. The server public key pk_S and the server secret key sk_S are given by executing $\mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$, and the receiver public key pk_R and the receiver secret key sk_R are given by executing $\mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$, where $\kappa \in \mathbb{N}$ is the security parameter. For a keyword ω, a trapdoor t_ω is given by executing $\mathsf{SCF\text{-}PEKS.Trapdoor}(sk_R, \omega)$, and a ciphertext λ is given by executing $\mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega)$. The server has a public/secret key pair (pk_S, sk_S), and a sender makes a ciphertext λ (which is encrypted by using a keyword ω', pk_S, and pk_R), and sends λ to the server. The server runs $\mathsf{SCF\text{-}PEKS.Test}(\lambda, sk_S, t_\omega)$, whose output is 1 if $\omega = \omega'$, and 0 otherwise. Note that obviously SCF-PEKS implies PEKS (i.e., if sk_S is publicly opened and (t_ω, sk_S) is regarded as a trapdoor of PEKS, then such a function-downgraded SCF-PEKS is PEKS). Correctness is defined as follows: For all $(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$, all $(pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$, and all $\omega \in \mathcal{K}$, $\mathsf{SCF\text{-}PEKS.Test}(\lambda, sk_S, t_\omega) = 1$ holds, where $\lambda \leftarrow \mathsf{SCF\text{-}PEKS.Enc}(pk_R, pk_S, \omega)$, $t_\omega \leftarrow \mathsf{SCF\text{-}PEKS.Trapdoor}(sk_R, \omega)$, and $\mathcal{K}$ is a keyword space.

Next, we consider two security requirements "consistency" and "keyword privacy".

Definition 8 (Consistency). *The SCF-PEKS scheme is said to be computationally consistent if the advantage is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv^{SCF\text{-}PEKS\text{-}CONSIST}_{\Pi,\mathcal{A}}(1^\kappa) = \Pr\big[(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa);\\
&\quad (pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa);\ (\omega, \omega') \leftarrow \mathcal{A}(pk_S, pk_R); \omega \neq \omega';\\
&\quad \lambda \leftarrow \mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega);\ t_{\omega'} \leftarrow \mathsf{SCF\text{-}PEKS.Trapdoor}(sk_R, \omega');\\
&\quad \mathsf{SCF\text{-}PEKS.Test}\big(\lambda, sk_S, t_{\omega'}\big) = 1\big]
\end{aligned}
$$

Next, we define two security notions for keyword privacy, "indistinguishability against chosen keyword attack with the server's secret key" (IND-CKA-SSK for short) and "indistinguishability against chosen keyword attack with all trapdoors" (IND-CKA-AT for short). In the IND-CKA-SSK experiment, an adversary $\mathcal{A}$ is assumed to be a malicious server. Therefore, $\mathcal{A}$ is given the server's secret key sk_S, whereas $\mathcal{A}$ cannot obtain the receiver's secret key sk_R. Instead of obtaining sk_R, $\mathcal{A}$ can issue a query to a trapdoor oracle $\mathcal{TRAP}$, which for an input keyword ω, returns a trapdoor t_ω. Note that $\mathcal{A}$ cannot query the challenge keywords ω_0^* and ω_1^* to $\mathcal{TRAP}$. As in the definition of [26], $\mathcal{A}$ computes (pk_S, sk_S), and gives pk_S to the challenger. So, we omit sk_S in the following experiment.

Definition 9 (IND-CKA-SSK). *An SCF-PEKS scheme is said to be IND-CKA-SSK-secure if the advantage is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv^{IND\text{-}CKA\text{-}SSK}_{\Pi,\mathcal{A}}(1^\kappa) =\\
&\quad \big|\Pr\big[(pk_S, State) \leftarrow \mathcal{A}(1^\kappa);\ (pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa);\\
&\qquad (\omega_0^*, \omega_1^*, State) \leftarrow \mathcal{A}^{\mathcal{TRAP}}(\mathsf{find}, pk_R, State);\ \mu \xleftarrow{\$} \{0,1\};\\
&\qquad \lambda^* \leftarrow \mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega_\mu^*); \mu' \leftarrow \mathcal{A}^{\mathcal{TRAP}}(\mathsf{guess}, \lambda^*, State);\\
&\qquad \mu = \mu'\big] - 1/2\big|
\end{aligned}
$$

Remark: Note that, for our TRE construction, the adversarial server's key generation above is not required. That is, the weaker definition can be used, where $\mathcal{C}$ can run $(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$, and sends (pk_S, sk_S) to $\mathcal{A}$ in our proof of Theorem 2.

Next, we define the adaptive-IND-CKA-AT experiment. In this experiment, an adversary $\mathcal{A}$ is assumed to be a malicious-but-legitimate receiver or outsider. Therefore, $\mathcal{A}$ is given the receiver's secret key sk_R, whereas $\mathcal{A}$ cannot obtain the server's secret key sk_S. This means that $\mathcal{A}$ knows *all* trapdoors. $\mathcal{A}$ can issue a query to a test oracle $\mathcal{TEST}$, which for an input (λ, t_ω) which satisfies $(\lambda, t_\omega) \notin \{(\lambda^*, t_{\omega_0^*}), (\lambda^*, t_{\omega_1^*})\}$, returns the result of the test algorithm. As in the definition of [26], $\mathcal{A}$ computes (pk_R, sk_R), and gives pk_R to the challenger. So, we omit sk_R in the following experiment.

Definition 10 (Adaptive-IND-CKA-AT). *An SCF-PEKS scheme is said to be adaptive-IND-CKA-AT-secure if the advantage is negligible for any PPT adversary $\mathcal{A}$ in the following experiment.*

$$
\begin{aligned}
&Adv^{Adaptive\text{-}IND\text{-}CKA\text{-}AT}_{\Pi,\mathcal{A}}(1^\kappa) = \\
&\quad \big| \Pr\big[(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa);\ (pk_R, State) \leftarrow \mathcal{A}(1^\kappa); \\
&\quad\ (\omega_0^*, \omega_1^*, State) \leftarrow \mathcal{A}^{\mathcal{TESST}}(\mathsf{find}, pk_S, State);\ \mu \xleftarrow{\$} \{0,1\}; \\
&\quad\ \lambda^* \leftarrow \mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega_\mu^*); \mu' \leftarrow \mathcal{A}^{\mathcal{TESST}}(\mathsf{guess}, \lambda^*, State); \\
&\quad\ \mu = \mu'\big] - 1/2\big|
\end{aligned}
$$

Remark: As in the IND-CKA-SSK, for TRE construction, the adversarial receiver's key generation above is not required. That is, we use the weaker definition, where $\mathcal{C}$ can run $(pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$, and sends (pk_R, sk_R) to $\mathcal{A}$ in our proof of Theorem 1.

4 Adaptive SCF-PEKS Implies TRE

4.1 Proposed TRE construction based on Adaptive SCF-PEKS

In this section, we propose a generic construction of TRE (with 1-bit plaintext space) based on adaptive SCF-PEKS. Our construction adopts the Boneh et al. IBE construction from PEKS [5], namely, for a plaintext 0 (resp. 1) and a release-time T, the time-release key is a trapdoor of the keyword $T||0$ (resp. $T||1$). In the following construction, a SCF-PEKS receiver is regarded as a TS, and a SCF-PEKS server is regarded as a TRE receiver. We set $\mathcal{T} = \mathcal{K}$ and $\mathcal{M}_{TRE} = \{0,1\}$.

Protocol 1 (TRE based on adaptive SCF-PEKS).

$\mathsf{TRE.Setup}(1^\kappa)$ *: Run* $(pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$*, set* $\mathsf{prm} = pk_R$ *and* $\mathsf{msk} = sk_R$*, and return* prm *and* msk*.*

$\mathsf{TRE.UKG}(1^\kappa)$ *: Run* $(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$*, set* $pk_u = pk_S$ *and* $sk_u = sk_S$*, and return* pk_u *and* sk_u*.*

$\mathsf{TRE.Ext}(\mathsf{prm}, \mathsf{msk}, T)$ *: Run* $t_{T0} \leftarrow \mathsf{SCF\text{-}PEKS.Trapdoor}(\mathsf{msk}, T||0)$ *and* $t_{T1} \leftarrow \mathsf{SCF\text{-}PEKS.Trapdoor}(\mathsf{msk}, T||1)$*, set* $s_T = (t_{T0}, t_{T1})$*, and return* s_T*.*

$\mathsf{TRE.Enc}(\mathsf{prm}, pk_u, T, M)$ *: For* $M \in \{0,1\}$*, run* $\lambda \leftarrow \mathsf{SCF\text{-}PEKS.Enc}(\mathsf{prm}, pk_u, T||M)$*, set* $C = \lambda$*, and return* C*.*

$\mathsf{TRE.Dec}(\mathsf{prm}, sk_u, s_T, C)$ *: Parse* $s_T = (t_{T0}, t_{T1})$*. If* $\mathsf{SCF\text{-}PEKS.Test}(C, sk_u, t_{T0}) = 1$ *holds, then output* $M = 0$*. Else if* $\mathsf{SCF\text{-}PEKS.Test}(C, sk_u, t_{T1}) = 1$ *holds, then output* $M = 1$*. Otherwise, output* $\perp$*.*

Obviously, correctness holds if the underlying SCF-PEKS satisfies correctness.

4.2 Security Analysis of our TRE construction

Theorem 1. *Our TRE construction satisfies IND-TR-CCA$_{TS}$ if the underlying SCF-PEKS satisfies adaptive IND-CKA-AT and consistency.*

Proof: Let $\mathcal{A}$ be an adversary who can break the IND-TR-CCA$_{\mathrm{TS}}$ security of our TRE construction, and $\mathcal{C}$ be the challenger of the adaptive IND-CKA-AT game. Then we construct an algorithm $\mathcal{B}$ which can break the adaptive IND-CKA-AT security (or consistency) of the underlying SCF-PEKS. First, $\mathcal{C}$ runs $(pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$ and $(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$, and sends (pk_R, sk_R, pk_S) to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{prm} = pk_R$, $\mathsf{msk} = sk_R$, and $pk_u = pk_S$, and sends $(\mathsf{prm}, \mathsf{msk}, pk_u)$ to $\mathcal{A}$.

Phase 1: For a decryption query (C, T), $\mathcal{B}$ issues two test queries $(C, T||0)$ and $(C, T||1)$ to $\mathcal{C}$. If $\mathcal{C}$ answers 0 for both queries, then $\mathcal{B}$ answers $\perp$. If $\mathcal{C}$ answers 1 for both queries, $\mathcal{B}$ can break consistency and aborts. Else, $\mathcal{C}$ answers 1 for the query $(C, T||M)$ $(M \in \{0,1\})$. Then $\mathcal{B}$ answers M.

Challenge: $\mathcal{A}$ sends (M_0^*, M_1^*, T^*) to $\mathcal{B}$. W.l.o.g, we set $M_0^* = 0$ and $M_1^* = 1$. $\mathcal{B}$ sends $(T^*||M_0^*, T^*||M_1^*) = (T^*||0, T^*||1)$ to $\mathcal{C}$ as the challenge keywords. $\mathcal{C}$ sends λ^* to $\mathcal{B}$. $\mathcal{B}$ sets $C^* = \lambda^*$, and sends C^* to $\mathcal{A}$. Note that C^* is a TRE ciphertext against either M_0^* or M_1^*.

Phase 2: For a decryption query $(C, T) \neq (C^*, T^*)$, $\mathcal{B}$ issues two test queries $(C, T||0)$ and $(C, T||1)$ to $\mathcal{C}$. If $\mathcal{C}$ answers 0 for both queries, then $\mathcal{B}$ answers $\perp$. If $\mathcal{C}$ answers 1 for both queries, $\mathcal{B}$ can break consistency and aborts. Else, $\mathcal{C}$ answers 1 for the query $(C, T||M)$ $(M \in \{0,1\})$. Then $\mathcal{B}$ answers M.

Guess: Finally, $\mathcal{A}$ outputs the guessing bit $\mu' \in \{0,1\}$. $\mathcal{B}$ outputs μ' as the guessing bit of the adaptive IND-CKA-AT game. □

Theorem 2. *Our TRE construction satisfies IND-TR-CPA$_{IS}$ if the underlying SCF-PEKS satisfies IND-CKA-SSK.*

Proof: Let $\mathcal{A}$ be an adversary who can break the IND-TR-CPA$_{\mathrm{IS}}$ security of our TRE construction, and $\mathcal{C}$ be the challenger of the IND-CKA-SSK game. Then we construct an algorithm $\mathcal{B}$ which can break the IND-CKA-SSK security of the underlying SCF-PEKS. First, $\mathcal{C}$ runs $(pk_R, sk_R) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$ and $(pk_S, sk_S) \leftarrow \mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$, and sends (pk_R, pk_S, sk_S) to $\mathcal{B}$. $\mathcal{B}$ sets $\mathsf{prm} = pk_R$, $pk_u = pk_S$, and $sk_u = sk_S$, and sends $(\mathsf{prm}, pk_u, sk_u)$ to $\mathcal{A}$.

Phase 1: For an extraction query T, $\mathcal{B}$ issues two trapdoor queries $T||0$ and $T||1$. $\mathcal{C}$ sends t_{T0} and t_{T1} to $\mathcal{B}$. $\mathcal{B}$ sets $s_T = (t_{T0}, t_{T1})$, and sends s_T to $\mathcal{A}$.

Challenge: $\mathcal{A}$ sends (M_0^*, M_1^*, T^*) to $\mathcal{B}$. $\mathcal{B}$ sends $(T^*||0, T^*||1)$ to $\mathcal{C}$ as the challenge keywords. $\mathcal{C}$ sends λ^* to $\mathcal{B}$. $\mathcal{B}$ sets $C^* = \lambda^*$, and sends C^* to $\mathcal{A}$. Note that C^* is a TRE ciphertext against either M_0^* or M_1^*.

Phase 2: For an extraction query $T \neq T^*$, $\mathcal{B}$ issues two trapdoor queries $T||0$ and $T||1$. $\mathcal{C}$ sends t_{T0} and t_{T1} to $\mathcal{B}$. $\mathcal{B}$ sets $s_T = (t_{T0}, t_{T1})$, and sends s_T to $\mathcal{A}$.

Guess: Finally, $\mathcal{A}$ outputs the guessing bit $\mu' \in \{0,1\}$. $\mathcal{B}$ outputs μ' as the guessing bit of the IND-CKA-SSK game. □

4.3 Discussion: The Reason Why PEKS and Non-Adaptive SCF-PEKS are not Suitable for Constructing TRE

First, we make it clear that we do not deny the possibility of TRE construction based on either PEKS or non-adaptive SCF-PEKS in the following discussion. But we observe that TRE requires two entities, called TS and receiver, and these entities have their public/secret key pair, respectively. So, it is hard to directly implement TRE from PEKS since PEKS requires just one entity (i.e., receiver). From the above considerations, SCF-PEKS is suitable for constructing TRE, since SCF-PEKS requires two entities, called server and receiver. Next, we need to implement the oracles defined in TRE security requirements in the SCF-PEKS context. The extraction query (in the IND-TR-CPA$_{\mathrm{IS}}$ experiment) can be implemented by the trapdoor oracle (IND-CKA-SSK) in the non-adaptive SCF-PEKS context. However, the decryption query (in the IND-TR-CCA$_{\mathrm{TS}}$ experiment) is hard to be implemented in the "non-adaptive" SCF-PEKS context, since no test query is considered in the IND-CKA-AT experiment. On the contrary, the decryption query can be handled by the test oracle in the adaptive SCF-PEKS context. This is the reason why we apply SCF-PEKS with adaptive security for constructing TRE. Note that although decryptable PEKS [12, 13, 17] might handle the decryption query, it requires just one entity, and therefore it is hard to directly implement TRE from decryptable PEKS. As a remark, IND-TR-CPA$_{\mathrm{TS}}$ secure TRE can be constructed from non-adaptive SCF-PEKS from the above considerations.

5 Anonymous IBE Implies Adaptive SCF-PEKS

5.1 Proposed Adaptive SCF-PEKS Construction

In this section, we give a generic construction of adaptive SCF-PEKS based on anonymous IBE, IND-stag-CCA TBE, and sUF OTS. In our construction, a ciphertext of an anonymous IBE scheme (say C_{IBE}) is used as a "plaintext" of a TBE scheme to hide keyword information from an adversary. From the result of the decryption of the TBE scheme, the ciphertext C_{IBE} must be obtained. In addition, usually, $C_{IBE} \notin \mathcal{M}_{TBE}$. To handle this condition, we apply the KEM/DEM framework [28] (a.k.a. hybrid encryption), where KEM stands for key encapsulation mechanism, and DEM stands for data encapsulation mechanism. By using TBE KEM (see Section 6 of [20]), compute $(K, C_{TBE}) \leftarrow \mathsf{TBE.Enc}(pk, t)$, and encrypt C_{IBE} as a plaintext of the CCA secure DEM such that $e = \mathsf{E}_K(C_{IBE})$. Note that a CCA-secure DEM can be generically constructed from any pseudorandom functions without redundancy. So, even if we assume that a CCA secure DEM exists, we do not need any additional cryptographic primitive, except anonymous IBE, for constructing adaptive SCF-PEKS. From now on, we assume that $C_{IBE} \in \mathcal{M}_{TBE}$ and $e = \mathsf{E}_K(C_{IBE})$ is implicitly included in C_{TBE} (i.e., C_{IBE} is obtained from the decryption of C_{TBE}).

In the following construction, we use a target collision resistant (TCR) hash function [3] $H_{tag} : \{0,1\}^* \to \mathcal{TAG}$.

Protocol 2 (Our Construction of Adaptive SCF-PEKS).

$\mathsf{SCF\text{-}PEKS.KeyGen_S}(1^\kappa)$**:** *Run* $(pk_S, sk_S) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$*, and output* (pk_S, sk_S)*.*

$\mathsf{SCF\text{-}PEKS.KeyGen_R}(1^\kappa)$**:** *Run* $(pk_R, sk_R) \leftarrow \mathsf{IBE.KeyGen}(1^\kappa)$*, and output* (pk_R, sk_R)*.*

$\mathsf{SCF\text{-}PEKS.Trapdoor}(sk_R, \omega)$**:** *Run* $t_\omega \leftarrow \mathsf{IBE.Extract}(sk_R, \omega)$*, and output* t_ω*.*

$\mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega)$**:** *Generate* $(K_s, K_v) \xleftarrow{\$} \mathsf{Sig.KeyGen}$*, compute* $t = H_{tag}(K_v)$*, choose* $R \xleftarrow{\$} \mathcal{M}_{IBE}$*, run* $C_{IBE} \leftarrow \mathsf{IBE.Enc}(pk_R, \omega, R)$*,* $C_{TBE} \leftarrow \mathsf{TBE.Enc}(pk_S, t, C_{IBE})$*, and* $\sigma \leftarrow \mathsf{Sign}(K_s, (C_{TBE}, R))$*, and output* $\lambda = (C_{TBE}, K_v, \sigma)$*.*

$\mathsf{SCF\text{-}PEKS.Test}(\lambda, sk_S, t_\omega)$**:** *Let* $\lambda = (C_{TBE}, K_v, \sigma)$*. Compute* $t = H_{tag}(K_v)$*, run* $C'_{IBE} \leftarrow \mathsf{TBE.Dec}(sk_S, t, C_{TBE})$ *and* $R' \leftarrow \mathsf{IBE.Dec}(t_\omega, C'_{IBE})$*. Output 1 if* $1{=}\mathsf{Verify}(K_v, \sigma, (C_{TBE}, R'))$*, and 0 otherwise.*

Obviously, correctness holds if the underlying TBE, IBE, and OTS satisfy correctness.

By observing our construction, non-adaptive SCF-PEKS (i.e., IND-CKA-AT without test queries, which has the same security requirement with Fang et al. [11]) can be constructed by reducing the one-time signature part and replacing the TBE part with CPA-secure PKE (i.e., chosen plaintext security is enough). A ciphertext is (C_{PKE}, R), where $C_{IBE} \leftarrow \mathsf{IBE.Enc}(pk_R, \omega, R)$ and $C_{PKE} \leftarrow \mathsf{PKE.Enc}(pk_S, C_{IBE})$. As in our adaptive SCF-PEKS construction, we assume that $C_{IBE} \in \mathcal{M}_{PKE}$, where $\mathcal{M}_{PKE}$ is the message space of the underlying PKE scheme. The test procedure is described as follows. Compute $C'_{IBE} \leftarrow \mathsf{PKE.Dec}(sk_S, C_{PKE})$ and $R' \leftarrow \mathsf{IBE.Dec}(t_\omega, C'_{IBE})$. Output 1 if $R' = R$, and 0 otherwise.

5.2 Security Analysis of our Adaptive SCF-PEKS construction

Theorem 3. *The SCF-PEKS scheme constructed by our method is computationally consistent if the underlying IBE scheme is IBE-IND-CPA secure.*

Proof: Let $\mathcal{A}$ be an adversary who breaks the computational consistency of SCF-PEKS constructed by the protocol 1, and $\mathcal{C}$ be the challenger of the IBE-IND-CPA experiment. Then, we can construct an algorithm $\mathcal{B}$ that breaks the IBE-IND-CPA security of the IBE scheme. First, $\mathcal{C}$ runs $\mathsf{IBE.Setup}(1^\kappa)$, and gives pk to $\mathcal{B}$. $\mathcal{B}$ sets pk as pk_R, runs $(pk_S, sk_S) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, and gives (pk_R, pk_S) to $\mathcal{A}$. $\mathcal{B}$ obtains keywords ω and ω' from $\mathcal{A}$. $\mathcal{B}$ chooses $R_0, R_1 \xleftarrow{\$} \mathcal{M}_{IBE}$ as the challenge messages, and sends (ω, R_0, R_1) to $\mathcal{C}$. $\mathcal{C}$ gives $C^*_{IBE} \leftarrow \mathsf{IBE.Enc}(pk_R, \omega, R_\mu)$ to $\mathcal{B}$, where $\mu \in \{0, 1\}$. $\mathcal{B}$ gets a trapdoor $t_{\omega'}$ by issuing an $\mathcal{EXTRACT}$ query. If $\mathsf{IBE.Dec}(t_{\omega'}, C^*_{IBE}) = R_1$, then $\mathcal{B}$ outputs 1, otherwise $\mathcal{B}$ outputs 0. □

Theorem 4. *The SCF-PEKS scheme constructed by our method is IND-CKA-SSK secure if the underlying IBE scheme is IBE-ANO-CPA secure.*

Proof: Let $\mathcal{A}$ be an adversary who breaks the IND-CKA-SSK security of SCF-PEKS constructed by the protocol 1, and $\mathcal{C}$ be the challenger of the IBE-ANO-CPA experiment. Then we can construct an algorithm $\mathcal{B}$ that breaks the IBE-ANO-CPA security of the underlying IBE scheme. First, $\mathcal{C}$ runs $\mathsf{IBE.Setup}(1^\kappa)$, and gives pk to $\mathcal{B}$. $\mathcal{B}$ sets pk as pk_R. $\mathcal{A}$ runs $(pk_S, sk_S) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, and gives pk_S to $\mathcal{B}$. For a $\mathcal{TRAP}$ query ω_i, $\mathcal{B}$ forwards ω_i to $\mathcal{C}$ as an $\mathcal{EXTRACT}$ query of the IBE scheme, gets t_{ω_i}, and answers t_{ω_i} to $\mathcal{A}$.

In the $\mathsf{Challenge}$ phase, $\mathcal{A}$ sends the challenge keywords ω_0^* and ω_1^* to $\mathcal{B}$. $\mathcal{B}$ chooses $R^* \xleftarrow{\$} \mathcal{M}_{IBE}$, and computes the challenge ciphertext as follows:

1. $\mathcal{B}$ sends $(R^*, \omega_0^*, \omega_1^*)$ to $\mathcal{C}$.
2. $\mathcal{C}$ gives $C_{IBE}^* \leftarrow \mathsf{IBE.Enc}(pk_R, \omega_\mu^*, R^*)$ to $\mathcal{B}$, where $\mu \in \{0,1\}$.
3. $\mathcal{B}$ generates $(K_s^*, K_v^*) \xleftarrow{\$} \mathsf{Sig.KeyGen}$, and computes $t^* = H_{tag}(K_v^*)$, $C_3^* \leftarrow \mathsf{TBE.Enc}(pk_S, t^*, C_{IBE}^*)$, and $\sigma^* \leftarrow \mathsf{Sign}(K_s^*, (C_{TBE}^*, R^*))$.
4. $\mathcal{B}$ sends $\lambda^* = (C_{TBE}^*, K_v^*, \sigma^*)$ to $\mathcal{A}$.

Note that $\mathcal{A}$ can compute $C_{IBE}^* \leftarrow \mathsf{TBE.Dec}(sk_S, H_{tag}(K_v^*), C_{TBE}^*)$. In addition, R^* may be revealed from σ^* without contradicting unforgeability property. However, this situation is the same as in the IBE-ANO-CPA experiment, where $\mathcal{A}$ inputs $ID_0^* := \omega_0^*$, $ID_1^* := \omega_1^*$, and $M^* := R^*$, and gets the challenge ciphertext C_{IBE}^*. Finally, $\mathcal{B}$ outputs μ', where $\mu' \in \{0,1\}$ is the output of $\mathcal{A}$. □

Theorem 5. *The SCF-PEKS scheme constructed by our method is adaptive-IND-CKA-AT secure if the underlying TBE scheme is IND-stag-CCA secure, the underlying signature is one-time sUF-CMA secure, and H_{tag} is a TCR hash function.*

Proof: Let $\mathcal{A}$ be an adversary who breaks the adaptive-IND-CKA-AT security of SCF-PEKS constructed by the protocol 1, and $\mathcal{C}$ be the challenger of the IND-stag-CCA experiment. Then, we can construct an algorithm $\mathcal{B}$ that breaks the IND-stag-CCA security of the underlying TBE scheme. First, $\mathcal{B}$ runs $(K_s^*, K_v^*) \leftarrow \mathsf{Sig.KeyGen}(1^\kappa)$, and sends $t^* := H_{tag}(K_v^*)$ to $\mathcal{C}$ as the challenge tag. $\mathcal{C}$ runs $\mathsf{TBE.KeyGen}(1^\kappa)$, and gives pk to $\mathcal{B}$. $\mathcal{B}$ sets pk as pk_S. $\mathcal{A}$ runs $(pk_R, sk_R) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, and gives pk_R to $\mathcal{B}$. Let $(\mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega_j) := (C_{TBE}, K_v, \sigma), t_{\omega_j})$ be a $\mathcal{TEST}$ query, where $\omega_j \in \mathcal{ID}$. $\mathcal{B}$ computes $t = H_{tag}(K_v)$, and answers as follows:

$t \neq t^*$: $\mathcal{B}$ can use the $\mathcal{DEC}$ oracle of the underlying TBE scheme as follows.
1. $\mathcal{B}$ forwards (C_{TBE}, t) to $\mathcal{C}$ as a $\mathcal{DEC}$ query of the TBE scheme.
2. $\mathcal{C}$ answers $C'_{IBE} \leftarrow \mathsf{TBE.Dec}(sk, t, C_{TBE})$.
   - Note that if t is not the legitimate tag of C_{TBE}, then $\mathcal{C}$ answers $\perp$. In this case, $\mathcal{B}$ answers 0.
3. $\mathcal{B}$ computes $R' \leftarrow \mathsf{IBE.Dec}(t_{\omega_j}, C'_{IBE})$.
4. If $\mathsf{Verify}(K_v, \sigma, (C_{TBE}, R')) = 1$, then $\mathcal{B}$ returns 1, and 0 otherwise.

$t = t^*$: If $K_v \neq K_v^*$, then $\mathcal{B}$ breaks the TCR property of H_{tag}. If $K_v = K_v^*$ (we call this a forge_1 event), then $\mathcal{B}$ gives a random answer in $\mathcal{C}$, and aborts.

In the $\mathsf{Challenge}$ phase, $\mathcal{A}$ sends the challenge keywords ω_0^* and ω_1^* to $\mathcal{B}$. $\mathcal{B}$ chooses $R^* \xleftarrow{\$} \mathcal{M}_{IBE}$, and computes the challenge ciphertext as follows:

1. $\mathcal{B}$ computes $C_{IBE,0} \leftarrow \mathsf{IBE.Enc}(pk_R, \omega_0^*, R^*)$ and $C_{IBE,1} \leftarrow \mathsf{IBE.Enc}(pk_R, \omega_1^*, R^*)$.
2. $\mathcal{B}$ sends $(M_0^*, M_1^*) := (C_{IBE,0}, C_{IBE,1})$ to $\mathcal{C}$ as the challenge messages of the IND-stag-CCA experiment of the TBE scheme.
3. $\mathcal{C}$ gives $C_{TBE}^* \leftarrow \mathsf{TBE.Enc}(pk_S, t^*, M_\mu^*)$ to $\mathcal{B}$.
4. $\mathcal{B}$ computes $\sigma^* \leftarrow \mathsf{Sign}(K_s^*, (C_{TBE}^*, R^*))$, and sends $\lambda^* = (C_{TBE}^*, K_v^*, \sigma^*)$ to $\mathcal{A}$.

Again, let $(\mathsf{SCF\text{-}PEKS.Enc}(pk_S, pk_R, \omega_j) := (C_{TBE}, K_v, \sigma), t_{\omega_j})$ be a $\mathcal{TEST}$ query, where $\omega_j \in \mathcal{ID}$. $\mathcal{B}$ computes $t = H_{tag}(K_v)$, and answers as follows:

In the case $t_{\omega_j} \in \{t_{\omega_0^*}, t_{\omega_1^*}\}$:

- $\boldsymbol{t = t^*}$: If $K_v \neq K_v^*$, then $\mathcal{B}$ breaks the TCR property of H_{tag}. If $K_v = K_v^*$ (we call this a forge_2 event), then $\mathcal{B}$ gives a random answer in $\mathcal{C}$, and aborts.
- $\boldsymbol{t \neq t^*}$: Then $\mathcal{B}$ can use the $\mathcal{DEC}$ oracle of the underlying TBE scheme as follows. .
  1. $\mathcal{B}$ forwards (C_{TBE}, t) to $\mathcal{C}$ as a $\mathcal{DEC}$ query of the TBE scheme.
  2. $\mathcal{C}$ answers $C'_{IBE} \leftarrow \mathsf{TBE.Dec}(sk, t, C_{TBE})$.
     - Note that if t is not the legitimate tag of C_{TBE}, then $\mathcal{C}$ answers $\perp$. In this case, $\mathcal{B}$ answers 0.
  3. $\mathcal{B}$ computes $R' \leftarrow \mathsf{IBE.Dec}(t_{\omega_j}, C'_{IBE})$.
  4. If $\mathsf{Verify}(K_v, \sigma, (C_{TBE}, R')) = 1$, then $\mathcal{B}$ returns 1, and 0 otherwise.

In the case $t_{\omega_j} \notin \{t_{\omega_0^*}, t_{\omega_1^*}\}$:

- $\boldsymbol{(C_{TBE}, K_v, \sigma) = (C_{TBE}^*, K_v^*, \sigma^*)}$: $\mathcal{B}$ returns 0, since $(C_{TBE}^*, K_v^*, \sigma^*)$ is an SCF-PEKS ciphertext of either ω_0^* or ω_1^*.
- $\boldsymbol{(C_{TBE}, K_v, \sigma) \neq (C_{TBE}^*, K_v^*, \sigma^*)}$: $\mathcal{B}$ runs the same simulation as in the find stage.

If $\mathcal{B}$ does not abort, then our simulation is perfect. Finally, $\mathcal{B}$ outputs μ', where $\mu' \in \{0, 1\}$ is the output of $\mathcal{A}$.

Next, we prove that $\Pr[\mathsf{forge}] := \Pr[\mathsf{forge}_1 \vee \mathsf{forge}_2]$ is negligible. We construct an algorithm $\mathcal{B}'$ which can win the sUF game with probability at least $\Pr[\mathsf{forge}]$. $\mathcal{B}'$ obtains K_v^* from the sUF challenger, instead of executing $\mathsf{Sig.KeyGen}(1^\kappa)$. $\mathcal{B}'$ runs $(pk_S, sk_S) \leftarrow \mathsf{TBE.KeyGen}(1^\kappa)$, and gives pk_S to $\mathcal{A}$. $\mathcal{A}$ runs $(pk_R, sk_R) \leftarrow \mathsf{IBE.Setup}(1^\kappa)$, and gives pk_R to $\mathcal{B}$. Since $\mathcal{B}'$ has sk_S, $\mathcal{B}'$ can answer any $\mathcal{TEST}$ queries. In the challenge phase of the adaptive-IND-CKA-AT experiment, $\mathcal{B}'$ computes $t^* = H_{tag}(K_v^*)$, chooses $R^* \xleftarrow{\$} \mathcal{M}_{IBE}$, runs $C_{IBE}^* \leftarrow \mathsf{IBE.Enc}(pk_R, \omega_\mu, R)$, and $C_{TBE}^* \leftarrow \mathsf{TBE.Enc}(pk_S, t^*, C_{IBE}^*)$, sets $M^* := (C_{TBE}^*, R^*)$, sends M^* to the sUF challenger, and obtains σ^* from the sUF challenger. Therefore, $\mathcal{B}'$ makes at most one signature query. Note that we do not have to care about the value $\mu \in \{0, 1\}$, since we only have to guarantee that $\lambda^* = (C_{TBE}^*, K_v^*, \sigma^*)$ is a valid SCF-PEKS ciphertext. In the forge events, $\mathcal{A}$ sends a $\mathcal{TEST}$ query $((C_{TBE}, K_v, \sigma), t_{\omega_j})$ with $K_v = K_v^*$.

forge_1 : In this case, $\mathcal{B}'$ can obtain a signature without issuing the signature query. $\mathcal{B}'$ computes $C_{IBE} \leftarrow \mathsf{TBE.Dec}(sk_S, H_{tag}(K_v), C_{TBE})$ and $R' \leftarrow \mathsf{IBE.Dec}(t_{\omega_j}, C_{IBE})$. If $((C_{TBE}, R'), \sigma)$ is not a valid signature pair, then $\mathcal{B}'$ returns 0 as the answer of this $\mathcal{TEST}$ query. Otherwise, if $((C_{TBE}, R'), \sigma)$ is a valid signature pair, then $\mathcal{B}'$ submits a forged pair $((C_{TBE}, R'), \sigma)$ to the sUF challenger and wins.

forge_2 : Now $t_{\omega_j} \in \{t_{\omega_0^*}, t_{\omega_1^*}\}$. Then $(C_{TBE}, \sigma) \neq (C^*_{TBE}, \sigma^*)$. $\mathcal{B}'$ computes $C_{IBE} \leftarrow \mathsf{TBE.Dec}(sk_S, H_{tag}(K_v), C_{TBE})$ and $R' \leftarrow \mathsf{IBE.Dec}(t_{\omega_j}, C_{IBE})$. If $((C_{TBE}, R'), \sigma)$ is not a valid signature pair, then $\mathcal{B}'$ returns 0 as the answer of this $\mathcal{TEST}$ query. Otherwise, if $((C_{TBE}, R'), \sigma)$ is a valid signature pair, then $\mathcal{B}'$ submits a forged pair $((C_{TBE}, R'), \sigma)$ to the sUF challenger and wins.

Therefore, $\Pr[\mathsf{forge}] := \Pr[\mathsf{forge}_1 \vee \mathsf{forge}_2]$ is negligible, since the underlying signature is sUF. □

6 Conclusion

In this paper, to show the relationships of IBE, TRE, and adaptive SCF-PEKS, we propose a generic construction of TRE with 1-bit plaintext space (resp. adaptive SCF-PEKS) from adaptive SCF-PEKS (resp. anonymous IBE). Our first result seems interesting since no bridge between TRE and PEKS primitive has been known before. In addition, no generic construction of SCF-PEKS has been proposed so far. That is, our second construction also seems independently interesting.

As future works, it is interesting to consider the keyword guessing attacks [18, 29], namely, if adaptive SCF-PEKS can handle keyword guessing attack, then what happens in the TRE context. In addition, we expect that the wildcard searching capability [27] might lead to a construction of time-specific encryption [25], where the time "interval" can be specified. Finally, a construction of TRE with multi-bit plaintext space from adaptive SCF-PEKS needs to be revisited.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. J. Cryptology 21(3), 350–391 (2008)
2. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: ICCSA (1). pp. 1249–1259 (2008)
3. Bellare, M., Rogaway, P.: Collision-resistant hashing: Towards making UOWHFs practical. In: CRYPTO. pp. 470–484 (1997)
4. Bellare, M., Shoup, S.: Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In: Public Key Cryptography. pp. 201–216 (2007)

5. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: EUROCRYPT. pp. 506–522 (2004)
6. Canetti, R., Halevi, S., Katz, J.: Chosen-ciphertext security from identity-based encryption. In: EUROCRYPT. pp. 207–222 (2004)
7. Cathalo, J., Libert, B., Quisquater, J.J.: Efficient and non-interactive timed-release encryption. In: ICICS. pp. 291–303 (2005)
8. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Provably secure timed-release public key encryption. ACM Trans. Inf. Syst. Secur. 11(2) (2008)
9. Cui, Y., Fujisaki, E., Hanaoka, G., Imai, H., Zhang, R.: Formal security treatments for IBE-to-signature transformation: Relations among security notions. IEICE Transactions 92-A(1), 53–66 (2009)
10. Dent, A.W., Tang, Q.: Revisiting the security model for timed-release encryption with pre-open capability. In: ISC. pp. 158–174 (2007)
11. Fang, L., Susilo, W., Ge, C., Wang, J.: A secure channel free public key encryption with keyword search scheme without random oracles. In: CANS. pp. 248–258 (2009)
12. Fang, L., Wang, J., Ge, C., Ren, Y.: Decryptable public key encryption with keyword search schemes. JDCTA 4(9), 141–150 (2010)
13. Fuhr, T., Paillier, P.: Decryptable searchable encryption. In: ProvSec. pp. 228–236 (2007)
14. Fujioka, A., Okamoto, Y., Saito, T.: Generic construction of strongly secure timed-release public-key encryption. In: ACISP. pp. 319–336 (2011)
15. Gu, C., Zhu, Y.: New efficient searchable encryption schemes from bilinear pairings. International Journal of Network Security 10(1), 25–31 (2010)
16. Gu, C., Zhu, Y., Pan, H.: Efficient public key encryption with keyword search schemes from pairings. In: Inscrypt. pp. 372–383 (2007)
17. Hofheinz, D., Weinreb, E.: Searchable encryption with decryption in the standard model. Cryptology ePrint Archive, Report 2008/423 (2008), http://eprint.iacr.org/
18. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Computer Communications 32(2), 394–396 (2009)
19. Khader, D.: Public key encryption with keyword search based on k-resilient IBE. In: ICCSA (3). pp. 1086–1095 (2007)
20. Kiltz, E.: Chosen-ciphertext security from tag-based encryption. In: TCC. pp. 581–600 (2006)
21. Matsuda, T., Nakai, Y., Matsuura, K.: Efficient generic constructions of timed-release encryption with pre-open capability. In: Pairing. pp. 225–245 (2010)
22. May, T.C.: Time-release crypto. Unpublished manuscript (1993)
23. Myers, S., Shelat, A.: Bit encryption is complete. In: FOCS. pp. 607–616 (2009)
24. Nakai, Y., Matsuda, T., Kitada, W., Matsuura, K.: A generic construction of timed-release encryption with pre-open capability. In: IWSEC. pp. 53–70 (2009)
25. Paterson, K.G., Quaglia, E.A.: Time-specific encryption. In: SCN. pp. 1–16 (2010)
26. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Improved searchable public key encryption with designated tester. In: ASIACCS. pp. 376–379 (2009)
27. Sedghi, S., van Liesdonk, P., Nikova, S., Hartel, P.H., Jonker, W.: Searching keywords with wildcards on encrypted data. In: SCN. pp. 138–153 (2010)
28. Shoup, V.: Using hash functions as a hedge against chosen ciphertext attack. In: EUROCRYPT. pp. 275–288 (2000)
29. Yau, W.C., Heng, S.H., Goi, B.M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: ATC. pp. 100–105 (2008)