Generic Constructions of Secure-Channel Free Searchable Encryption with Adaptive Security *

Keita Emura† Atsuko Miyaji† Mohammad Shahriar Rahman§ Kazumasa Omote¶

December 27, 2016

Abstract

For searching keywords against encrypted data, the public key encryption scheme with keyword search (PEKS), and its an extension called secure-channel free PEKS (SCF-PEKS) have been proposed. In SCF-PEKS, a receiver makes a trapdoor for a keyword, and uploads it on a server. A sender computes an encrypted keyword, and sends it to the server. The server executes the searching procedure (called the test algorithm, which takes as inputs an encrypted keyword, trapdoor, and secret key of the server). In this paper, we extend the security of SCF-PEKS, calling it adaptive SCF-PEKS, wherein an adversary (modeled as a “malicious-but-legitimate” receiver) is allowed to issue test queries adaptively, and show that adaptive SCF-PEKS can be generically constructed by anonymous identity-based encryption (anonymous IBE) only. That is, for constructing adaptive SCF-PEKS we need not require any additional cryptographic primitive when compared to the Abdalla et al. PEKS construction (J. Cryptology 2008), even though adaptive SCF-PEKS requires additional functionalities. Note that our generic construction needs to apply the KEM/DEM framework (a.k.a. hybrid encryption), where KEM stands for key encapsulation mechanism, and DEM stands for data encapsulation mechanism. We also show that there is a class of anonymous IBE that can be applied for constructing adaptive SCF-PEKS without using hybrid encryption, and propose an adaptive SCF-PEKS construction based on this IBE. Although our second construction is not fully generic, it is efficient compared to the first, since we can exclude the DEM part. Finally, we instantiate an adaptive SCF-PEKS scheme (via our second construction) that achieves a similar level of efficiency for the costs of the test procedure and encryption, compared to the (non-adaptive secure) SCF-PEKS scheme by Fang et al. (CANS2009).

Keywords : Public-key Encryption with Keyword Search, Adaptive Security, Anonymous Identity-Based Encryption

1 Introduction

1.1 Public Key Encryption Scheme with Keyword Search (PEKS)

PEKS was proposed by Boneh et al. [9], and considers searching keywords from encrypted data. Briefly, the flow of PEKS is as follows: A receiver makes a trapdoor $t_\omega$ for a keyword $\omega$, and uploads it on a server. A

---

*Preliminary versions of this paper appear in the 14th Information Security Conference, ISC 2011 [20], and 7th International Conference on Security and Cryptography, SECRYPT 2012 [23]. The first half (the generic construction part) is based on [20], and the second half (not fully generic but more efficient construction part) is based on [23]. This is the full and the merged version. We also corrected some typos appeared in the journal version [21].

†National Institute of Information and Communications Technology (NICT), Japan. k-emura@nict.go.jp
‡School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Japan. miyaji@jaist.ac.jp
§Department of Computer Science and Engineering, University of Asia Pacific, Bangladesh. shahriar.rahman@uap-bd.edu
¶School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Japan. omote@jaist.ac.jp
sender makes a ciphertext of a keyword $\omega'$ by using the receiver public key, and sends it to the server. The server outputs 1 if $\omega = \omega'$, by using $t_\omega$, and 0 otherwise.

PEKS was investigated from both theoretical and practical perspectives. For example, several PEKS schemes with additional functionality have been proposed thus far: PEKS schemes treating plural keywords [11, 33, 41, 46], PEKS with public key encryption (PEKS/PKE) [3, 50], a decryptable PEKS scheme [26], where a receiver can decrypt an encrypted keyword, and a public key encryption with an oblivious keyword search scheme [14], wherein the server can obtain trapdoors without revealing the keywords. From the theoretical perspective, a PEKS scheme based on Jacobi symbols has been proposed [17] (though almost all PEKS schemes are constructed by using bilinear maps). The off-line keyword guessing attack was also introduced in [13, 34, 43, 48, 31], wherein an adversary can guess what keywords were used for computing trapdoors. The notion of interactive PEKS, wherein the trapdoor is generated interactively by the sender and the receiver, has also been proposed [16]. Moreover, PEKS with perfect keyword privacy has been considered in [40] which treats the leakage of keywords from trapdoors. PEKS with trapdoor revocation has also been considered in [22, 38, 49].

As a feasibility result of PEKS, Abdalla et al. [1] showed that a generic construction of PEKS based on anonymous IBE is sufficient.

1.2 Security Conditions of Previous Secure-Channel Free PEKS (SCF-PEKS) Schemes and the Theoretic Extension

PEKS schemes ensure that the server (or an outsider) learns nothing about keywords chosen by the sender without trapdoor information. Namely, 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 the channel requires additional setup costs. To solve this problem, secure-channel free PEKS (SCF-PEKS) has been proposed [4, 24, 29, 30, 35], wherein the server has a public/secret key pair, and the sender makes a ciphertext of a keyword $\omega'$ (which is encrypted by the server public key and the receiver public key), and sends it to the server. The server outputs 1 if $\omega = \omega'$ by using the trapdoor $t_\omega$ and its own secret key, and 0 otherwise. Even if $t_\omega$ 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 models of the previous SCF-PEKS schemes [4, 24, 29, 30, 35] do not capture the test queries (i.e., no adversary can issue test queries in security games). We point out that this definition does not capture the real environment as follows. Fig.1 illustrates how to instantiate test queries in the real world.

1. A malicious receiver computes (or eavesdrops on) a trapdoor, and uploads it to the server.
   - From the viewpoint of the server, this is the same as uploading a trapdoor from a valid receiver.
2. The malicious receiver computes (or eavesdrops on) an SCF-PEKS ciphertext, and sends it to the server.
   - This is the same as sending a ciphertext from a valid sender.

3. The malicious receiver can obtain the result of the test algorithm.

Through the above procedure, the malicious receiver can obtain the result of the test algorithm. In other words, the malicious receiver can use the server as the test oracle. SCF-PEKS therefore has to be secure, even if the malicious receiver can be admitted to issue test queries. The test queries were considered in [42], but this definition is still weak (i.e., “Unquoted CCA-like” security [39]), since test queries cannot be repeated adaptively as follows: Let $\lambda^* = [A_1^*, A_2^*, \ldots, A_n^*]$ be the challenge ciphertext and $t_{\omega^0_0}$ (resp. $t_{\omega^1_1}$) be the trapdoors corresponding to the challenge keyword $\omega^0_0$ (resp. $\omega^1_1$). In the definition of [42], $\mathcal{A}$ is allowed to issue $(\lambda, t_{\omega})$ such that $t_{\omega} \notin \{t_{\omega^0_0}, t_{\omega^1_1}\}$ and for all $i \in [1, n]$, $A_i \neq A_i^*$. This is not natural, since $\mathcal{A}$ may compute a ciphertext $\lambda$ and replace a part of $\lambda$ with a part of $\lambda^*$. This scenario can easily be handled in the above real world example by sending such a “replaced ciphertext” in the guess phase. By considering the CCA2 security, SCF-PEKS must be secure even if a “malicious-but-legitimate” receiver can be admitted to issue test queries adaptively. We insist that this adaptive (i.e., “CCA2-like”) security is theoretically the natural extension of the SCF-PEKS security, which is called adaptive SCF-PEKS.

1.3 Our Contribution

In this paper, 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. Note that IND-stag-CCA-secure TBE can be constructed by selective-ID chosen plaintext (sID-CPA) secure IBE [36], and the digital signature can be constructed by IBE [18]. Therefore, our result shows that adaptive SCF-PEKS can be constructed by anonymous IBE only. That is, we show that for constructing adaptive SCF-PEKS no additional cryptographic primitive is required when compared to the Abdalla et al. PEKS construction [1], even though adaptive SCF-PEKS requires additional functionalities. This construction uses double encryption, wherein TBE encrypts a ciphertext of an anonymous IBE. Since the ciphertext space of IBE is usually not equal to the plaintext space of TBE, we apply the KEM/DEM framework [45] (a.k.a. hybrid encryption), where KEM stands for key encapsulation mechanism, and DEM stands for data encapsulation mechanism.

Next, we show that there is a class of anonymous IBE that can be applied for constructing adaptive SCF-PEKS without using hybrid encryption, and propose an adaptive SCF-PEKS construction based on such an IBE. Although it is not fully generic, our second construction is efficient compared to the first one since we can exclude the DEM part.

Finally, we instantiate an adaptive SCF-PEKS scheme based on the Gentry anonymous IBE scheme [27], the Kiltz IND-stag-CCA-secure TBE scheme [36], and the Bellare-Shoup sUF OTS scheme [7], by using our second adaptive SCF-PEKS construction. Our concrete adaptive SCF-PEKS construction (called the GKBS construction due to the author’s name) achieves a similar level of efficiency for the costs of the test procedure and encryption, compared to the (non-adaptive secure) SCF-PEKS scheme without random oracles proposed by Fang et al [24] (see the comparison table (Table 2) in section 6.3).

Remark. Independent of our result, Fang et al. [25] proposed a concrete SCF-PEKS scheme with keyword guessing attack resilience. They also considered the test oracle, and give a formal security definition. The efficiency of our concrete instantiation (GKBS) and that of the Fang et al. SCF-PEKS scheme is almost similar. Later, Guo and Yan [31] proposed more efficient SCF-PEKS scheme secure in the Fang et al. model. Note that no generic construction is given in [25, 31], and therefore proposing a generic construction of adaptive SCF-PEKS with keyword guessing attack resilience is an interesting future work.
2 Preliminaries

This section, we define the building tools for our generic adaptive SCF-PEKS construction. $x \in S$ means that $x$ 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 $\text{State}$ as the state information transmitted by the adversary to himself across stages of the attack in experiments.

### 2.1 Definitions of IND-stag-CCA Secure TBE

In the following, $T_{\mathcal{A}}$ and $M_{\text{TBE}}$ are a tag space of TBE and a plaintext space of TBE, respectively.

**Definition 2.1 (Syntax of TBE).** A TBE scheme $\Pi$ consists of the following three algorithms, $\text{TBE.KeyGen}$, $\text{TBE.Enc}$ and $\text{TBE.Dec}$:

- **$\text{TBE.KeyGen}(1^\kappa)$** : This algorithm takes as an input the security parameter $\kappa \in \mathbb{N}$, and returns a public key $pk$ and a secret key $sk$.

- **$\text{TBE.Enc}(pk, t, M)$** : This algorithm takes as inputs $pk$, a message $M \in M_{\text{TBE}}$ with a tag $t \in T_{\mathcal{A}}$, and returns a ciphertext $C_{\text{TBE}}$.

- **$\text{TBE.Dec}(sk, t, C_{\text{TBE}})$** : This algorithm takes as inputs $sk$, $t$, and $C_{\text{TBE}}$, and returns $M$ or $\bot$.

Correctness is defined as follows: For all $(pk, sk) \leftarrow \text{TBE.KeyGen}(1^\kappa)$, all $M \in M_{\text{TBE}}$, and all $t \in T_{\mathcal{A}}$,

$$\text{TBE.Dec}(sk, t, C_{\text{TBE}}) = M$$

Next, we define the security requirement of TBE under selective-tag CCA (IND-stag-CCA) as follows.

**Definition 2.2 (IND-stag-CCA).** For any PPT adversary $A$ and the security parameter $\kappa \in \mathbb{N}$, we define the experiment $\text{Exp}_{\Pi, A}^{\text{IND-stag-CCA}}(1^\kappa)$ in Figure 2, and define the advantage of $A$ $\text{Adv}_{\Pi, A}^{\text{IND-stag-CCA}}(1^\kappa)$ as follows.

$$\text{Adv}_{\Pi, A}^{\text{IND-stag-CCA}}(1^\kappa) := \left| \Pr \left[ \text{Exp}_{\Pi, A}^{\text{IND-stag-CCA}}(1^\kappa) \right] - \frac{1}{2} \right|$$

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

A TBE scheme $\Pi$ is said to be IND-stag-CCA secure if the advantage $\text{Adv}_{\Pi, A}^{\text{IND-stag-CCA}}(1^\kappa)$ is negligible.

### 2.2 Definitions of Anonymous IBE

In the following, $ID$ and $M_{\text{IBE}}$ are an identity space and a plaintext space of IBE, respectively.

**Definition 2.3 (Syntax of IBE).** IBE scheme $\Pi$ consists of the following four algorithms, $\text{IBE.Setup}$, $\text{IBE.Extract}$, $\text{IBE.Enc}$ and $\text{IBE.Dec}$:
Next, we define anonymity experiment of IBE under CPA (IBE-ANO-CPA).

IBE-ANO-CPA

\[ \text{Adv}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) := \left| \Pr \left[ \text{Exp}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) \right] - \frac{1}{2} \right| \]

Here, \( \text{EXTRACT} \) is the extraction oracle for input of an identity \( ID \) it returns the corresponding secret key \( sk_{ID} \). Note that \( ID^* \) is not allowed as input to \( \text{EXTRACT} \) in the IBE-IND-CPA experiment.

An IBE scheme \( \Pi \) is said to be IBE-IND-CPA secure if the advantage \( \text{Adv}_{\Pi}^{\text{IBE-IND-CPA}}(1^\kappa) \) is negligible.

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

Definition 2.5 (IBE-ANO-CPA). For any PPT adversary \( \mathcal{A} \) and the security parameter \( \kappa \in \mathbb{N} \), we define the experiment \( \text{Exp}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) \) in Figure 3, and define the advantage of \( \mathcal{A} \) \( \text{Adv}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) \) as follows.

\[ \text{Adv}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) := \left| \Pr \left[ \text{Exp}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) \right] - \frac{1}{2} \right| \]

\( ID_0^* \) and \( ID_1^* \) are not allowed as input to \( \text{EXTRACT} \) in the IBE-ANO-CPA experiment. An IBE scheme \( \Pi \) is said to be IBE-ANO-CPA secure if the advantage \( \text{Adv}_{\Pi}^{\text{IBE-ANO-CPA}}(1^\kappa) \) is negligible.

Definition 2.6 (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.
2.3 Definitions of sUF OTS

In the following, \( \mathcal{M}_{\text{Sig}} \) is a message space of OTS.

**Definition 2.7 (Syntax of OTS).** A strongly existentially unforgeable (sUF) OTS against adaptively chosen message attack (CMA) (e.g., [7]) consists of the following three algorithms, \( \text{Sig, KeyGen, Sign} \) and \( \text{Verify} \):

\[
\text{Sig}(K_s; M) : \quad \text{This algorithm takes as inputs } K_s \text{ and a message } M \in \mathcal{M}_{\text{Sig}}, \text{ and returns a signature } \sigma.
\]

\[
\text{Verify}(K_v; \sigma; M) : \quad \text{This algorithm takes as inputs } K_v, \sigma, \text{ and } M, \text{ and returns } 1 \text{ if } \sigma \text{ is a valid signature of } M, \text{ and } 0 \text{ otherwise.}
\]

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

**Definition 2.8 (one-time sUF-CMA).** For any PPT adversary \( A \) and the security parameter \( \kappa \in \mathbb{N} \), we define the experiment \( \text{Exp}_{\Pi, A}^{\text{one-time sUF-CMA}}(1^\kappa) \) in Figure 4, and define the advantage of \( A \) \( \text{Adv}_{\Pi, A}^{\text{one-time sUF-CMA}}(1^\kappa) \) as follows:

\[
\text{Adv}_{\Pi, A}^{\text{one-time sUF-CMA}}(1^\kappa) := \Pr \left[ \text{Exp}_{\Pi, A}^{\text{one-time sUF-CMA}}(1^\kappa) \right]
\]

A signature scheme \( \Pi \) is said to be one-time sUF-CMA secure if the advantage \( \text{Adv}_{\Pi, A}^{\text{one-time sUF-CMA}}(1^\kappa) \) is negligible.

3 Definitions of Adaptive SCF-PEKS

In this section, we define security requirements of SCF-PEKS. In the following, \( \mathcal{K} \) is a keyword space.

**Definition 3.1 (Syntax of SCF-PEKS).** An SCF-PEKS scheme \( \Pi \) consists of the following five algorithms, \( \text{SCF-PEKS.KeyGen}_S, \text{SCF-PEKS.KeyGen}_R, \text{SCF-PEKS.Trapdoor, SCF-PEKS.Enc and SCF-PEKS.Test} \):

\[
\text{SCF-PEKS.KeyGen}_S(1^\kappa) : \quad \text{This server key generation algorithm takes as input the security parameter } 1^\kappa \text{ (} \kappa \in \mathbb{N} \text{), and returns a server public key } pk_S \text{ and a server secret key } sk_S.
\]
Definition 3.2

Exp_{\Pi, A}^{SCF-PEKS-CONSIST}(1^\kappa) := [(pk_S, sk_S) \leftarrow SCF-PEKS.KeyGen_S(1^\kappa); (pk_R, sk_R) \leftarrow SCF-PEKS.KeyGen_R(1^\kappa); 
(\omega, \omega') \leftarrow A(pk_S, pk_R); \omega \neq \omega'; \lambda \leftarrow SCF-PEKS.Enc(pk_S, pk_R, \omega); t_\omega \leftarrow SCF-PEKS.Trapdoor(sk_R, \omega'); \overline{SCF-PEKS.Test}(\lambda, sk_S, t_\omega) = 1]

IND-CKA-SSK

Exp_{\Pi, A}^{IND-CKA-SSK}(1^\kappa) := [(pk_S, State) \leftarrow A(1^\kappa); (pk_R, sk_R) \leftarrow SCF-PEKS.KeyGen_R(1^\kappa); 
(\omega'_0, \omega'_1, State) \leftarrow A^{\text{TRAP}}(\text{find}, pk_R, State); \mu \in \{0, 1\}; \lambda' \leftarrow SCF-PEKS.Enc(pk_S, pk_R, \omega'_\mu); \mu' \leftarrow A^{\text{TEST}}(\text{guess}, \lambda', State); \mu = \mu']

Adaptive-IND-CKA-AT

Exp_{\Pi, A}^{Adaptive-IND-CKA-AT}(1^\kappa) := [(pk_S, sk_S) \leftarrow SCF-PEKS.KeyGen_S(1^\kappa); (pk_R, State) \leftarrow A(1^\kappa); 
(\omega'_0, \omega'_1, State) \leftarrow A^{\text{TEST}}(\text{find}, pk_R, State); \mu \in \{0, 1\}; \lambda' \leftarrow SCF-PEKS.Enc(pk_S, pk_R, \omega'_\mu); \mu' \leftarrow A^{\text{TEST}}(\text{guess}, \lambda', State); \mu = \mu']

SCF-PEKS.KeyGen_R(1^\kappa) : This receiver key generation algorithm takes as input the security parameter 1^\kappa (\kappa \in \mathbb{N}), and returns a receiver public key pk_R and a receiver secret key sk_R.

SCF-PEKS.Trapdoor(sk_R, \omega) : This trapdoor generation algorithm takes as input sk_R and a keyword \omega \in \mathcal{K}, and returns a trapdoor t_\omega corresponding to keyword \omega.

SCF-PEKS.Enc(pk_S, pk_R, \omega) : This encryption algorithm takes as input pk_R, pk_S, and \omega, and returns a ciphertext \lambda.

SCF-PEKS.Test(\lambda, sk_S, t_\omega) : This test algorithm takes as input \lambda, sk_S, and t_\omega, and returns 1 if \omega = \omega', where \omega' is the keyword which was used for computing \lambda, and 0 otherwise.

A sender makes a ciphertext \lambda of a keyword \omega' using both pk_S and pk_R, and sends \lambda to the server. The server runs SCF-PEKS.Test(\lambda, sk_S, t_\omega), whose output is 1 if \omega = \omega', and 0 otherwise.

We require the correctness property as follows: For all (pk_S, sk_S) \leftarrow SCF-PEKS.KeyGen_S(1^\kappa), all (pk_R, sk_R) \leftarrow SCF-PEKS.KeyGen_R(1^\kappa), and all \omega \in \mathcal{K}, SCF-PEKS.Test(\lambda, sk_S, t_\omega) = 1 holds, where \lambda \leftarrow SCF-PEKS.Enc(pk_R, pk_S, \omega) and t_\omega \leftarrow SCF-PEKS.Trapdoor(\omega, sk_R).

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

Definition 3.2 (Consistency). For any PPT adversary A and the security parameter \kappa \in \mathbb{N}, we define the experiment Exp_{\Pi, A}^{SCF-PEKS-CONSIST}(1^\kappa) as follows.

\text{Adv}_{\Pi, A}^{SCF-PEKS-CONSIST}(1^\kappa) := \Pr \left[ Exp_{\Pi, A}^{SCF-PEKS-CONSIST}(1^\kappa) \right]

The SCF-PEKS scheme \Pi is said to be computationally consistent if the advantage Adv_{\Pi, A}^{SCF-PEKS-CONSIST}(1^\kappa) is negligible.
assumed to be a malicious server. Therefore, \( \mathcal{A} \) is given the server secret key \( sk_S \), whereas \( \mathcal{A} \) cannot obtain the receiver secret key \( sk_R \). Instead of obtaining \( sk_R \), \( \mathcal{A} \) can issue a query to a trapdoor oracle \( \mathcal{T R A P} \), which is defined in Table 1. As in the definition of [42], \( \mathcal{A} \) computes \((pk_S, sk_S)\), and gives \( pk_S \) to the challenger. So, we omit \( sk_S \) in the IND-CKA-SSK experiment.

**Definition 3.3 (IND-CKA-SSK).** For any PPT adversary \( \mathcal{A} \) and the security parameter \( \kappa \in \mathbb{N} \), we define the experiment \( \text{Exp}_{\Pi, \mathcal{A}}^{\text{IND-CKA-SSK}}(1^\kappa) \) in Figure 5, and define \( \text{Adv}_{\Pi, \mathcal{A}}^{\text{IND-CKA-SSK}}(1^\kappa) \) as follows.

\[
\text{Adv}_{\Pi, \mathcal{A}}^{\text{IND-CKA-SSK}}(1^\kappa) := \left| \Pr \left[ \text{Exp}_{\Pi, \mathcal{A}}^{\text{IND-CKA-SSK}}(1^\kappa) \right] - \frac{1}{2} \right|
\]

An SCF-PEKS scheme \( \Pi \) is said to be IND-CKA-SSK-secure if the advantage \( \text{Adv}_{\Pi, \mathcal{A}}^{\text{IND-CKA-SSK}}(1^\kappa) \) is negligible.

**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 \( C \) can run \((pk_S, sk_S) \leftarrow \text{SCF-PEKS.KeyGen}^\kappa(1^\kappa)\), 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 secret key \( sk_R \), whereas \( \mathcal{A} \) cannot obtain the server secret key \( sk_S \). This means that \( \mathcal{A} \) knows all trapdoors. \( \mathcal{A} \) can issue a query to a test oracle \( \text{TEST} \), which is defined in Table 1. As in the definition of [42], \( \mathcal{A} \) computes \((pk_R, sk_R)\), and gives \( pk_R \) to the challenger. So, we omit \( sk_R \) in the Adaptive-IND-CKA-AT experiment.

**Definition 3.4 (Adaptive-IND-CKA-AT).** For any PPT adversary \( \mathcal{A} \) and the security parameter \( \kappa \in \mathbb{N} \), we define the experiment \( \text{Exp}_{\Pi, \mathcal{A}}^{\text{Adaptive-IND-CKA-AT}}(1^\kappa) \) in Figure 5, and define \( \text{Adv}_{\Pi, \mathcal{A}}^{\text{Adaptive-IND-CKA-AT}}(1^\kappa) \) as follows.

\[
\text{Adv}_{\Pi, \mathcal{A}}^{\text{Adaptive-IND-CKA-AT}}(1^\kappa) := \left| \Pr \left[ \text{Exp}_{\Pi, \mathcal{A}}^{\text{Adaptive-IND-CKA-AT}}(1^\kappa) \right] - \frac{1}{2} \right|
\]

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.

**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 \( C \) can run \((pk_R, sk_R) \leftarrow \text{SCF-PEKS.KeyGen}^\kappa(1^\kappa)\), and sends \((pk_R, sk_R) \) to \( \mathcal{A} \) in our proof of Theorem 1.

### 4 Anonymous IBE Implies Adaptive SCF-PEKS

#### 4.1 A Generic Construction of Adaptive SCF-PEKS

This section gives 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 \( C_{\text{IBE}} \)) 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 \( C_{\text{IBE}} \) must be obtained. In addition, usually, \( C_{\text{IBE}} \notin \mathcal{M}_{\text{TBE}} \). To handle this condition, we apply the KEM/DEM framework [45] (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 [36]), compute \((K, C_{\text{TBE}}) \leftarrow \text{TBE.Enc}(pk, t)\), and encrypt \( C_{\text{IBE}} \) as a plaintext of the CCA secure DEM such that \( e = E_K(C_{\text{IBE}}) \). Note that a CCA-secure DEM can be generically constructed from any pseudorandom functions without redundancy [37]. So, even if we assume that a CCA secure DEM exists, we need no additional cryptographic primitive, except anonymous IBE, for constructing adaptive SCF-PEKS. From here, we assume that \( C_{\text{IBE}} \in \mathcal{M}_{\text{TBE}} \) and \( e = E_K(C_{\text{IBE}}) \) is implicitly included in \( C_{\text{TBE}} \) (i.e., \( C_{\text{IBE}} \) is obtained from the decryption of \( C_{\text{TBE}} \)). Note that section 5 gives an extended construction (which does not require hybrid encryption).
In the following construction, as in the Abdalla et al. PEKS construction [1], a keyword \( \omega \) is regarded as an “identity” of IBE. As in the Kiltz CCA-secure PKE construction [36] based on IND-stag-CCA TBE, a verification key \( K_v \) is regarded as a “tag” of TBE. We use a target collision-resistant (TCR) hash function [6] \( H_{tag} : \{0, 1\}^* \rightarrow \mathcal{G} \). We set \( \mathcal{M}_{tag} = \mathcal{C}_{TBE} \times \mathcal{M}_{IBE} \), where \( \mathcal{C}_{TBE} \) is a ciphertext space of the underlying TBE.

**Our First Adaptive SCF-PEKS Construction**

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

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

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

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

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

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

Intuitively, an adversary (in the IND-CKA-SSK experiment) that has the server secret key (i.e., a decryption key of TBE) can compute \( C_{IBE} \) from \( C_{TBE} \). However, since such an adversary does not have trapdoors to the challenge keywords, no information about keywords leaks from a ciphertext of an anonymous IBE, even if \( R \) is revealed from \( \sigma \) (without contradicting unforgeable property). In other words, we can reduce from the IND-CKA-SSK experiment to the IBE-ANO-CPA experiment. TBE is applied to hide information about keywords from an adversary (in the adaptive-IND-CKA-AT experiment) that has all the trapdoors. In other words, the adversary loses the opportunity to apply trapdoors to challenge keywords to the challenge ciphertext. In addition, due to the sUF property of the underlying signature scheme and the TCR property of \( H_{tag} \), the adversary cannot issue a test query that the simulator cannot answer.

**Non-adaptive SCF-PEKS construction:** By observing our construction, a non-adaptive SCF-PEKS (i.e., IND-CKA-AT without test queries, which has the same security requirement as Fang et al. [24]) 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 \( C_{IBE} \leftarrow \text{IBE.Enc}(pk_R, \omega, R) \) and \( C_{PKE} \leftarrow \text{PKE.Enc}(pk_S, C_{IBE}) \). 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 \( C'_{IBE} \leftarrow \text{PKE.Dec}(sk_S, C_{PKE}) \), and \( R' \leftarrow \text{IBE.Dec}(t_\omega, C'_{IBE}) \). Output 1 if \( R' = R \), and 0 otherwise.

### 4.2 Security Analysis of our First Adaptive SCF-PEKS construction

**Theorem 4.1.** The SCF-PEKS scheme constructed by our method is computationally consistent if the underlying IBE scheme is IBE-IND-CPA secure.

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

In the Challenge phase, $A$ sends the challenge keywords $\omega_0^*$ and $\omega_1^*$ to $B$. $B$ chooses $R^* \leftarrow \mathcal{M}_{IBE}$, and computes the challenge ciphertext as follows:

1. $B$ sends $(R^*, \omega_0^*, \omega_1^*)$ to $C$.
2. $C$ gives $C^*_{IBE} \leftarrow \text{IBE.Enc}(pk_R, \omega_0^*, R^*)$ to $B$, where $u \in \{0, 1\}$.
3. $B$ generates $(K_i^*, \sigma_i^*) \leftarrow \text{Sig.KeyGen}(1^k)$, and computes $t^* = H_{tag}(K_i^*)$, $C^*_{TBE} \leftarrow \text{TBE.Enc}(pk_S, t^*, C^*_BE)$, and $\sigma^* \leftarrow \text{Sign}(K^*_TBE, (C^*_TBE, R^*))$.
4. $B$ sends $\lambda^* = (C^*_TBE, K_i^*, \sigma^*)$ to $A$.

Note that $A$ can compute $C^*_{IBE} \leftarrow \text{TBE.Dec}(sk_B, H_{tag}(K_i^*), C^*_TBE)$. In addition, $R^*$ may be revealed from $\sigma^*$ without contradicting unforgeability property. However, this situation is the same as in the IBE-ANO-CPA experiment, where $A$ inputs $ID^0_0 = \omega_0^*$, $ID^1_0 = \omega_1^*$, and $M^* := R^*$, and gets the challenge ciphertext $C^*_BE$. Finally, $B$ outputs $\mu'$, where $\mu' \in \{0, 1\}$ is the output of $A$.

Theorem 4.3. 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 $A$ be an adversary who breaks the adaptive-IND-CKA-AT security of SCF-PEKS constructed by the protocol 1, and $C$ be the challenger of the IND-stag-CCA experiment. Then, we can construct an algorithm $B$ that breaks the IND-stag-CCA security of the underlying TBE scheme. First, $B$ runs $(K_i^*, \sigma_i^*) \leftarrow \text{Sig.KeyGen}(1^k)$, and sends $t^* := H_{tag}(K_i^*)$ to $C$ as the challenge tag. $C$ runs $\text{TBE.KeyGen}(1^k)$, and gives $pk$ to $B$. $B$ sets $pk$ as $pk_S$. $A$ runs $(pk_R, sk_R) \leftarrow \text{IBE.Setup}(1^k)$, and gives $pk_R$ to $B$. Let $(\text{SCF-PEKS.Enc}(sk_B, pk_R, \omega_1) := (C^*_TBE, K_i, \sigma_i, t_{\omega_1})$ be a $\text{TEST}$ query, where $\omega_1 \in \mathcal{T}_D$. $B$ computes $t = H_{tag}(K_i)$, and answers as follows:

$t \neq t^*$ : $B$ can use the $\text{DEC}$ oracle of the underlying TBE scheme as follows.

1. $B$ forwards $(C^*_TBE, t)$ to $C$ as a $\text{DEC}$ query of the TBE scheme.
2. $C$ answers $C^*_{IBE} \leftarrow \text{TBE.Dec}(sk_B, C^*_TBE)$.
   
   - Note that if $t$ is not the legitimate tag of $C^*_TBE$, then $C$ answers $\perp$. In this case, $B$ answers $0$.
3. $B$ computes $R' \leftarrow \text{IBE.Dec}(t_{\omega_1}, C^*_IBE)$.
4. If $\text{Verify}(K_i, \sigma_i, (C^*_TBE, R')) = 1$, then $B$ returns $1$, and $0$ otherwise.

$t = t^*$ : If $K_i \neq K_i^*$, then $B$ breaks the TCR property of $H_{tag}$. If $K_i = K_i^*$ (we call this a forge$_1$ event), then $B$ gives a random answer in $\mathcal{C}$, and aborts.

In the Challenge phase, $A$ sends the challenge keywords $\omega_0^*$ and $\omega_1^*$ to $B$. $B$ chooses $R^* \leftarrow \mathcal{M}_{IBE}$, and computes the challenge ciphertext as follows:

1. $B$ computes $C_{IBE,0}^* \leftarrow \text{IBE.Enc}(pk_R, \omega_0^*, R^*)$ and $C_{IBE,1}^* \leftarrow \text{IBE.Enc}(pk_R, \omega_1^*, R^*)$.
2. $B$ sends $(M^*_0, M^*_1) := (C_{IBE,0}^*, C_{IBE,1}^*)$ to $C$ as the challenge messages of the IND-stag-CCA experiment of the TBE scheme.
3. \( C \) gives \( C_{TBE}^* \leftarrow \text{TBE.Enc}(pk_S, t^*, M'_\mu) \) to \( B \).

4. \( B \) computes \( \sigma^* \leftarrow \text{Sign}(K_v^*, (C_{TBE}^*, R^*)) \), and sends \( \lambda^* = (C_{TBE}^*, K_v^*, \sigma^*) \) to \( A \).

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

**In the case \( t_{\omega_j} \in \{t_{\omega_0}, t_{\omega_1}\} \):**

- \( t = t^* \): If \( K_v \neq K_v^* \), then \( B \) breaks the TCR property of \( H_{tag} \). If \( K_v = K_v^* \) (we call this a \text{forge}_2 event), then \( B \) gives a random answer in \( C \), and aborts.

- \( t \neq t^* \): Then \( B \) can use the \( \mathcal{DEC} \) oracle of the underlying TBE scheme as follows.

  1. \( B \) forwards \((C_{TBE}, t) \) to \( C \) as a \( \mathcal{DEC} \) query of the TBE scheme.
  2. \( C \) answers \( C'_{IBE} \leftarrow \text{TBE.Dec}(sk, t, C_{TBE}) \).

     • Note that if \( t \) is not the legitimate tag of \( C_{TBE} \), then \( C \) answers \( \perp \). In this case, \( B \) answers 0.
  3. \( B \) computes \( R' \leftarrow \text{IBE.Dec}(t_{\omega_j}, C'_{IBE}) \).
  4. If \( \text{Verify}(K_v, \sigma, (C_{TBE}, R')) = 1 \), then \( B \) returns 1, and 0 otherwise.

**In the case \( t_{\omega_j} \notin \{t_{\omega_0}, t_{\omega_1}\} \):**

\((C_{TBE}, K_v, \sigma) = (C_{TBE}^*, K_v^*, \sigma^*) \): \( B \) returns 0, since \((C_{TBE}^*, K_v^*, \sigma^*) \) is an SCF-PEKS ciphertext of either \( \omega_0^* \) or \( \omega_1^* \).

\((C_{TBE}, K_v, \sigma) \neq (C_{TBE}^*, K_v^*, \sigma^*) \): \( B \) runs the same simulation as in the find stage.

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

Next, we prove that \( \Pr[\text{forge}] := \Pr[\text{forge}_1 \lor \text{forge}_2] \) is negligible. We construct an algorithm \( B' \) which can win the sUF game with probability at least \( \Pr[\text{forge}] \). \( B' \) obtains \( K_v^* \) from the sUF challenger, instead of executing \( \text{Sig.KeyGen}(1^n) \). \( B' \) runs \((pk_S, sk_S) \leftarrow \text{TBE.KeyGen}(1^n) \), and gives \( pk_S \) to \( A \). \( A \) runs \((pk_R, sk_R) \leftarrow \text{IBE.Setup}(1^n) \), and gives \( pk_R \) to \( B \). Since \( B' \) has \( sk_S, B' \) can answer any \( \mathcal{TEST} \) queries. In the challenge phase of the adaptive-IND-CKA-AT experiment, \( B' \) computes \( t^* = H_{tag}(K_v^*) \), chooses \( R^* \leftarrow \mathcal{M}_{IBE} \), runs \( C'_{IBE} \leftarrow \text{IBE.Enc}(pk_R, \omega_\mu, R) \), and \( C_{TBE}^* \leftarrow \text{TBE.Enc}(pk_S, t^*, C'_{IBE}) \), sets \( M^* := (C_{TBE}^*, R^*) \), sends \( M^* \) to the sUF challenger, and obtains \( \sigma^* \) from the sUF challenger. Therefore, \( B' \) 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, \( A \) sends a \( \mathcal{TEST} \) query \(((C_{TBE}, K_v, \sigma, t_{\omega_j}) \) with \( K_v = K_v^* \).

**\text{forge}_1:** In this case, \( B' \) can obtain a signature without issuing the signature query. \( B' \) computes \( C_{IBE} \leftarrow \text{TBE.Dec}(sk_S, H_{tag}(K_v^*), C_{TBE}) \) and \( R' \leftarrow \text{IBE.Dec}(t_{\omega_j}, C_{IBE}) \). If \(((C_{TBE}, R'), \sigma) \) is not a valid signature pair, then \( B' \) returns 0 as the answer of this \( \mathcal{TEST} \) query. Otherwise, if \(((C_{TBE}, R'), \sigma) \) is a valid signature pair, then \( B' \) submits a forged pair \(((C_{TBE}, R'), \sigma) \) to the sUF challenger and wins.

**\text{forge}_2:** Now \( t_{\omega_j} \in \{t_{\omega_0}, t_{\omega_1}\} \). Then \((C_{TBE}, \sigma) \neq (C_{TBE}^*, \sigma^*) \). \( B' \) computes \( C_{IBE} \leftarrow \text{TBE.Dec}(sk_S, H_{tag}(K_v), C_{TBE}) \) and \( R' \leftarrow \text{IBE.Dec}(t_{\omega_j}, C_{IBE}) \). If \(((C_{TBE}, R'), \sigma) \) is not a valid signature pair, then \( B' \) returns 0 as the answer of this \( \mathcal{TEST} \) query. Otherwise, if \(((C_{TBE}, R'), \sigma) \) is a valid signature pair, then \( B' \) submits a forged pair \(((C_{TBE}, R'), \sigma) \) to the sUF challenger and wins.

Therefore, \( \Pr[\text{forge}] := \Pr[\text{forge}_1 \lor \text{forge}_2] \) is negligible, since the underlying signature is sUF. ☐
4.3 IBE with Partitioned Ciphertext Structure (PCS-IBE)

The role of the KEM/DEM framework in the first adaptive SCF-PEKS construction (presented in section 4) is that an IBE ciphertext is regarded as a TBE plaintext to hide keyword information from an adversary that has $mk$ (this adversary appears in the Adaptive-IND-CKA-AT experiment). In this section, we propose an extension of the first SCF-PEKS construction, which need not require hybrid encryption. Unfortunately, we assume that the underlying IBE belongs to a special class, however, while previously known anonymous IBE schemes [10, 12, 14, 15, 19, 27, 44] belong to this class. In other words, we do not have to apply hybrid encryption as long as a previously known anonymous IBE scheme (enumerated in the above list) is used as a building tool of adaptive SCF-PEKS.

Here, we show that the KEM/DEM framework can be reduced if the underlying IBE satisfies the following properties (called IBE with partitioned ciphertext structure (PCS-IBE)\(^1\)).

**Definition 4.1 (PCS-IBE).** IBE is said to be PCS-IBE if its ciphertext $C_{IBE}$ can be split into two parts $C_{IBE} := (C_{IBE,1}, C_{IBE,2})$ with the following properties.

- $C_{IBE,1}$ is a single group element.
  - Note that the essential condition is $C_{IBE,1} \in \mathcal{M}_{TBE}$. However, since Kiltz [36] proposed a TBE scheme with $\mathcal{M}_{TBE} = \mathcal{G}$ and $\mathcal{M}_{TBE} = \mathcal{G}_T$, respectively, where $(\mathcal{G}, \mathcal{G}_T)$ is a bilinear group, it is enough to require that $C_{IBE,1}$ be a single group element.

- $C_{IBE,1}$ only includes an identity ID (i.e., $C_{IBE,2}$ is independent of ID).

- For any common message $M$ and distinct identities $ID$ and $ID'$ ($ID \neq ID'$), $C_{IBE,2}$ can be commonly used for $(C_{IBE,1}', C_{IBE,2}) \leftarrow \text{IBE.Enc}(pk, ID, M; s)$ and $(C_{IBE,1}, C_{IBE,2}') \leftarrow \text{IBE.Enc}(pk, ID', M; s)$ if the same random number $s$ is used for both encryptions.
  - That is, both $(C_{IBE,1}, C_{IBE,2})$ and $(C_{IBE,1}', C_{IBE,2}')$ are valid ciphertexts.

This structure is used for computing the challenge ciphertext in the proof of the adaptive IND-CKA-AT. In the proof, no matter which plaintext $(C_{IBE,1}, C_{IBE,1}')$ is encrypted, both $C_{IBE,2}$ and $C_{IBE,2}'$ can be used as a part of the challenge ciphertext, since $C_{IBE,2} = C_{IBE,2}'$ due to the PCS property.

Here, we explain the above structure in the Boneh-Franklin (BF) IBE scheme [10] case due to its easy-to-understand structure as follows: For a message $M$ and an identity $ID$, a ciphertext $(C_{IBE,1}, C_{IBE,2})$ is described as $C_{IBE,1} = M \cdot e(Y, \text{Hash}(ID)^s)$ and $C_{IBE,2} = g^s$, where $Y$ is a public key of the key authority, and $s \in \mathbb{Z}_p$ (with a prime order $p$) is a random number of an encryptor’s choice. Then, for the common message $M$, an another identity $ID'$, and the same random number $s$, $(C_{IBE,1}', C_{IBE,2}')$ is also a valid ciphertext, where $C_{IBE,1}' = M \cdot e(Y, \text{Hash}(ID')^s)$.

In the Gentry IBE case (which is used for our instantiations), for a message $M$ and an identity $ID$, a ciphertext $(C_{IBE,1}, C_{IBE,2})$ is described as $C_{IBE,1} = (g^g)^{-ID} \cdot M$ and $C_{IBE,2} = e(g, g)^s \cdot M \cdot e(g, h)^{-s}$. So, for the common message $M$, an another identity $ID'$, and the same random number $s$, $(C_{IBE,1}', C_{IBE,2}')$ is also a valid ciphertext, where $C_{IBE,1}' = (g^g)^{-ID'} \cdot M$.

4.4 The Second Adaptive SCF-PEKS Construction based on PCS-IBE

Although SCF-PEKS.KeyGenS, SCF-PEKS.KeyGenR, and SCF-PEKS.Trapdoor are the same as these of the first adaptive SCF-PEKS construction (protocol 2), for the sake of clarity, we describe these algorithms in the following. Let a ciphertext space of the underlying PCS-IBE be $C_{IBE} = C_{IBE,1} \times C_{IBE,2}$. We set $C_{IBE,1} = \mathcal{M}_{TBE}$ and $\mathcal{M}_{Sig} = C_{IBE,2} \times C_{TBE} \times \mathcal{M}_{IBE}$.

\(^1\)Note that our partitioned requirement is different from that of partitioned IBKEM [2].

12
Our Second Adaptive SCF-PEKS Construction w/o Hybrid Encryption

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

SCF-PEKS.KeyGen$_B(1^k)$: Run $(pk_R, sk_R) \leftarrow \text{IBE.KeyGen}(1^k)$, and output $(pk_R, sk_R)$.

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

SCF-PEKS.Enc$(pk_S, pk_R, \omega)$: Generate $(K_s, K_v) \leftarrow \text{Sig.KeyGen}$, compute $t = H_{160}(K_v)$, choose $R \leftarrow \mathcal{M}_{\text{IBE}}$, run $(C_{\text{IBE},1}, C_{\text{IBE},2}) \leftarrow \text{IBE.Enc}(pk_R, \omega, R)$, $C_{\text{TBE}} \leftarrow \text{TBE.Enc}(pk_S, t, C_{\text{IBE},1})$, and $\sigma \leftarrow \text{Sign}(K_s, (C_{\text{IBE},2}, C_{\text{TBE}}, R))$, and output $C = (C_{\text{IBE},2}, C_{\text{TBE}}, K_v, \sigma)$.

SCF-PEKS.Test$(\lambda, sk_s, t_\omega)$: Let $\lambda = (C_{\text{IBE},2}, C_{\text{TBE}}, K_v, \sigma)$. Compute $t = H_{160}(K_v)$, and run $C'_{\text{IBE},1} \leftarrow \text{TBE.Dec}(sk_s, t, C_{\text{TBE}})$ and $R' \leftarrow \text{IBE.Dec}(t_\omega, (C'_{\text{IBE},1}, C_{\text{IBE},2}))$. Output 1 if $1=\text{Verify}(K_v, \sigma, (C_{\text{IBE},2}, C_{\text{TBE}}, R'))$, and 0 otherwise.

The security proofs are the same as those of the first ones, except the construction of the challenge ciphertext in the adaptive-IND-CKA-AT experiment as follows: In the Challenge phase, $B$ needs to compute the SCF-PEKS challenge ciphertext $\lambda^*$, although $B$ does not know $\mu \in \{0,1\}$ chosen by the TBE challenger $C$. So, our PCS property comes into effect to compute $\lambda^*$, since $B$ can use both $C_{\text{IBE},2}$ and $C_{\text{IBE},1}$ (so we set $C_{\text{IBE},2} = C_{\text{IBE},1}$). More concretely, in the Challenge phase, $A$ sends the challenge keywords $\omega_0$ and $\omega_1$ to $B$. $B$ chooses $R^* \leftarrow \mathcal{M}_{\text{IBE}}$ and computes the challenge ciphertext as follows:

1. $B$ computes $(C_{\text{IBE},1}, C_{\text{IBE},2}) \leftarrow \text{IBE.Enc}(pk_R, \omega_0^*, R^*)$ and $(C_{\text{IBE},1}, C_{\text{IBE},2}) \leftarrow \text{IBE.Enc}(pk_R, \omega_1^*, R^*)$ using the same random number (i.e., $C_{\text{IBE},2} = C_{\text{IBE},1}$). $B$ sets $C_{\text{IBE},2} := C_{\text{IBE},1}$.
2. Note that both $(C_{\text{IBE},1}, C_{\text{IBE},2})$ and $(C_{\text{IBE},1}, C_{\text{IBE},2})$ are valid ciphertexts of the underlying IBE scheme. This is the reason we require anonymous “PCS”-IBE.

3. $B$ sends $(M_0^*, M_1^*) := (C_{\text{IBE},1}, C_{\text{IBE},1})$ to $C$ as the challenge messages.

4. $C$ gives $C_{\text{TBE}} \leftarrow \text{TBE.Enc}(pk_S, t^*, M_\mu^*)$ to $B$, where $\mu \in \{0,1\}$ is the challenge bit.

4. $B$ computes $\sigma^* \leftarrow \text{Sign}(K_v^*, (C_{\text{IBE},2}, C_{\text{TBE}}, R^*))$, and sends $\lambda^* = (C_{\text{IBE},2}, C_{\text{TBE}}, K_v, \sigma^*)$ to $A$.

Then, $\lambda^*$ is a valid ciphertext due to the PCS property. Since $B$ does not have to consider the bit $\mu$ chosen by $C$, $B$ can use $C_{\text{IBE},2}$.

As in the first one, non-adaptive SCF-PEKS can be constructed by reducing the one-time signature part and replacing the TBE part with CPA-secure PKE. Let the underlying IBE be PCS, then a ciphertext is $(C_{\text{IBE},2}, C_{\text{PKE}}, R)$, where $(C_{\text{IBE},1}, C_{\text{IBE},2}) \leftarrow \text{IBE.Enc}(pk_R, \omega, R)$ and $C_{\text{PKE}} \leftarrow \text{PKE.Enc}(pk_S, C_{\text{IBE},1})$.

4.5 Comparison Between Our First/Second Adaptive SCF-PEKS Constructions

In the first adaptive SCF-PEKS construction (protocol 1, section 4), the DEM part $e = E_k(C_{\text{IBE}})$ is implicitly included in $C_{\text{TBE}}$. Here, we explicitly include $e$ in a SCF-PEKS ciphertext as follows: $\lambda_1 = (e, C_{\text{TBE}}, K_v, \sigma)$ (subscript 1 means that it is a ciphertext of the first adaptive SCF-PEKS construction). On the contrary, $\lambda_2 = (C_{\text{IBE},2}, C_{\text{TBE}}, K_v, \sigma)$ in the second SCF-PEKS construction (proposed in this section, protocol 2, subscript 2 means that it is a ciphertext of the second adaptive SCF-PEKS construction). Since the size of $e$ is at least the same size of $C_{\text{IBE}}$, by excluding the DEM part, the size of the ciphertext of the second construction (say $|\lambda_2|$) is smaller than that of the first one (say $|\lambda_1|$), i.e., $|\lambda_1| \geq |C_{\text{IBE},1}| + |\lambda_2|$. Since the ciphertext size is the most bottlenecked point of our adaptive SCF-PEKS construction compared to the concrete constructions, we can say that the second adaptive SCF-PEKS construction is more efficient than the first, though it is not fully generic.
Table 2: Comparison between our constructions and the Fang et al. SCF-PEKS

<table>
<thead>
<tr>
<th></th>
<th>Comp. (\lambda)</th>
<th>Comp. Test</th>
<th>Length of (\lambda)</th>
<th>Adaptive Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fang et al. [24]</td>
<td>(2ME(G) + 3ME(G_T)) ((11ME(G)))</td>
<td>(ME(G) + 2ME(G_T) + 2BM) ((7ME(G) + 2BM))</td>
<td>(2</td>
<td>G</td>
</tr>
<tr>
<td>GBBS construction</td>
<td>(4ME(G) + 2ME(G_T)) ((10ME(G)))</td>
<td>(ME(G) + ME(G_T) + BM) ((4ME(G) + BM))</td>
<td>(3</td>
<td>G</td>
</tr>
<tr>
<td>GKBS construction</td>
<td>(8ME(G) + 2ME(G_T)) ((14ME(G)))</td>
<td>(5ME(G) + ME(G_T) + BM) ((8ME(G) + BM))</td>
<td>(7</td>
<td>G</td>
</tr>
</tbody>
</table>

5 A Concrete Instantiation of Adaptive SCF-PEKS

5.1 The GKBS Construction

Here, by using the extended version of our adaptive SCF-PEKS construction, we instantiate an adaptive SCF-PEKS scheme based on the Gentry (PCS) anonymous IBE [27], the Kiltz IND-stag-CCA-secure TBE [36], and the Bellare-Shoup sUF one-time signature [7]. We call it the GKBS construction by picking up the authors' name. Let \(G\) and \(G_T\) be cyclic groups of prime order \(p, e\) an efficiently computable bilinear map \(\epsilon : G \times G \rightarrow G_T\), and \(H_{sig} : [0,1]^{\kappa} \times [0,1]^* \rightarrow Z_p\) be a CR hash function, where each \(\kappa\)-bit key \(K\) specifies a particular hash function \(H(K,\cdot)\) with domain \([0,1]^*\). We assume that \(e(g, g)\) and \(e(g, h)\) are included in public keys to reduce the number of pairing computations.

An adaptive SCF-PEKS scheme without random oracles (the GKBS construction)

**SCF-PEKS.KeyGen**(1\(^\kappa\)): Choose \(g_1 \leftarrow G\) and \(x_1, x_2, y_1, y_2 \leftarrow Z_p\). Choose \(g_2, z \in G\) with \(g_1^{x_1} = g_2^{x_2} = z\). Compute \(u_1 = g_1^{x_1}\) and \(u_2 = g_2^{x_2}\). Output ((\(pk_S, sk_s\)) = ((\(g_1, g_2, z, u_1, u_2, (x_1, x_2, y_1, y_2)\)).

**SCF-PEKS.KeyGen**(1\(^\kappa\)): Choose \(g, h \leftarrow G\) and \(\alpha \leftarrow Z_p\), compute \(g' = g^\alpha\), and output ((\(pk_R, sk_r\)) = ((\(g', h, e(g, g), e(g, h), \alpha\)).

**SCF-PEKS.Trapdoor**(\(sk_R, \omega\)): For a keyword \(\omega \in Z_p\), choose \(r_\omega \leftarrow Z_p\), compute \(h_\omega = (hg^{-r_\omega})^{\frac{1}{e}}\), and output \(t_\omega = (r_\omega, h_\omega)\).

**SCF-PEKS.Enc**(\(pk_S, pk_R, \omega\)): Choose \(R \leftarrow G_T\), \(s, r_1, r_2, x, y \leftarrow Z_p\), and \(K \leftarrow [0,1]^{\kappa}\). Compute \(X = g_R^x, Y = g^y\), set \(K_v = (K, X, Y)\), and compute \(t = H_{tag}(K_v)\), \(C_{IBE_1} = (g^{y - \epsilon})^s, C_{IBE_2} = (e(g, g)^s, R, e(g, h)^{-\omega})\), \(C_{TBE} = (g_1^{x_1}g_2^{x_2}(z^\omega u_1)^{r_1}, (z^\omega u_2)^{r_2}, C_{IBE_1} \cdot z^{x_1 + r_1})\), \(c = H_{sig}(K, Y)(|C_{IBE_2}, C_{TBE}, R)\), and \(\sigma = c + yr \mod p\). Output \(\lambda = (C_{IBE_2}, C_{TBE}, \sigma, K_v)\).

**SCF-PEKS.Test**(\(\lambda, sk_s, t_\omega\)): Parse \(sk_s = (x_1, x_2, y_1, y_2), t_\omega = (r_\omega, h_\omega), C_{IBE_2} = (f_1, f_2), C_{TBE} = (v_1, v_2, v_3, v_4, v_5)\), and \(K_v = (K, X, Y)\). Compute \(t = H_{tag}(K_v)\), and check \(v_1^{x_1 + r_1} \equiv v_3\) and \(v_2^{x_2 + r_2} \equiv v_4\). If not, then output 0. Otherwise, compute \(C_{IBE_1} = v_5/(v_1^{x_1} \cdot v_2^{x_2})\), \(R' = f_1^\omega \cdot e(C_{IBE_1}, h_\omega) \cdot f_2\), and \(c = H_{sig}(K, Y)(|C_{IBE_2}, C_{TBE}, R')\), and check \(g^\omega = YX^\omega\). If not, then output 0. Otherwise, output 1.

We assume the difficulty of the one-more-discrete-log (omdl) problem [5]\(^2\), the decisional augmented bilinear Diffie-Hellman exponent (decisional ABDHE) problem [27], and the gap decision linear (gap DLIN) problem [36], and the collision resistance of \(H_{tag}\) and \(H_{sig}\). Then, the above SCF-PEKS instantiation is adaptive secure in the standard model.

\(^2\)We can use a discrete-log-based sUF one-time signature [28, 47] with one more \(Z_p\) element.
Next, we estimate the efficiency of the GKBS construction. Although concrete SCF-PEKS schemes have been proposed [4, 29, 30, 42], these schemes are proved in the random oracle model. As a well-known fact, efficient cryptographic schemes can be constructed easily if the random oracle is assumed. So, we focus on SCF-PEKS schemes proposed by Fang et al. [24] and Khader [35], respectively, which are secure in the standard model. Khader [35] shows that PEKS and SCF-PEKS can be constructed by using $k$-resilient IBE [32] (which is an IBE scheme, wherein an adversary can obtain at most $k$ private keys of IDs). Since $k$-resilient IBE [32] is designed by applying a DDH-hard group without pairings, the Khader PEKS/SCF-PEKS also enables pairing-free constructions. Unfortunately, the Khader PEKS/SCF-PEKS schemes require $k$-dependent large number of public keys and high encryption costs. Therefore, here we compare our GKBS construction to the Fang et al. SCF-PEKS scheme [24] in Table 2 (the Fang et al. SCF-PEKS scheme is introduced in the Appendix). In addition, for comparison, we instantiate a non-adaptive SCF-PEKS scheme (using the second construction). We call this the GBBS construction which is based on the Gentry IBE scheme [27] and the linear encryption scheme presented by Boneh, Boyen, and Shacham [8] (the actual construction of this non-adaptive SCF-PEKS scheme is given in the Appendix). The GBBS construction achieves the same security level as that of the Fang et al. construction.

Let $ME(G)$ and $ME(G_T)$ be the computational costs of multi-exponentiation in $G$ and $G_T$, respectively, $BM$ be that of one bilinear map computation, and $|G|$, $|G_T|$, and $|\mathbb{Z}_p|$ be the bit-length of the representation of an element of $G$, $G_T$, and $\mathbb{Z}_p$, respectively. More precisely, we assume that the security parameter $\kappa = 170$. Therefore, $p$ is a 170-bit prime, $|G| = 171$ bits and $|G_T| = 1020$ bits: i.e., we assume that $G$ is an elliptic curve defined over finite field $\mathbb{F}_p$ and $G_T$ is a multiplicative group on finite field $\mathbb{F}_p^\times$ with the embedded degree $k = 6$. In this case, the computational complexity over $G_T$ is approximately three times higher than that of $G$. So, we estimate $ME(G_T) = 3ME(G)$, and write them in Table 2 in parentheses. Although in the GKBS construction the length of the ciphertext is larger than that of the Fang et al. construction, the computation of the Test algorithm is faster (if $BM < ME(G)$ which usually holds). Therefore, there is not much efficiency difference between our GKBS construction and the Fang et al. scheme, although the GKBS construction enables adaptive security.

6 Conclusion

In this paper, from a theoretical perspective, we show that no additional cryptographic primitive is required compared to the Abdalla et al. PEKS construction, even though adaptive SCF-PEKS requires additional functionalities.

From a practical perspective, since malicious receivers can use the server as the test oracle, our adaptive security notion is applicable in practice. In addition, our concrete adaptive SCF-PEKS construction (the GKBS construction) achieves a similar level of efficiency for the costs of the test procedure and encryption, compared to the (non-adaptive secure) SCF-PEKS scheme without random oracles proposed by Fang et al, even though adaptive SCF-PEKS requires additional functionalities.

References


Here, we introduce the SCF-PEKS scheme proposed by Fang et al. [24]. Let $G$ and $G_T$ be cyclic groups of prime order $p$, $e$ be an efficiently computable bilinear map $e: G \times G \rightarrow G_T$ and $g \in G$ be a generator. We assume that $e(X, Q), e(g, g),$ and $e(g, h)$ are included in public keys to reduce pairing computations.
The Fang et al. SCF-PEKS scheme [24]

SCF-PEKS.KeyGen$_5$(1$^*$): Choose $x \xleftarrow{\$} Z_p$ and $Q \xleftarrow{\$} G$, compute $X = g^x$, and output $(pk_S, sk_S) = ((X, Q, e(X, Q), e(g, g)), x)$.

SCF-PEKS.KeyGen$_R$(1$^*$): Choose $y \xleftarrow{\$} Z_p$ and $h \xleftarrow{\$} G$, compute $Y = g^y$, and output $(pk_R, sk_R) = ((Y, h, e(g, h)), y)$.

SCF-PEKS.Trapdoor$(sk_R, \omega)$: For a keyword $\omega \in Z_p$, choose $r_\omega \xleftarrow{\$} Z_p$, compute $h_\omega = (hg^{-r_\omega})^{\frac{1}{x_z}}$, and output $t_\omega = (r_\omega, h_\omega)$.

SCF-PEKS.Enc$(pk_S, pk_R, \omega)$: For a keyword $\omega$, choose $s, r \xleftarrow{\$} Z_p$, compute $C_1 = g^s$, $t = H(e(C_1, Q)^s)$, $C_2 = (Yg^{-\omega})^{r/t}$, $C_3 = e(g, g)^r$, and $C_4 = e(g, h)^r$, and output $\lambda = (C_1, C_2, C_3, C_4)$.

SCF-PEKS.Test$(\lambda, sk_S, t_\omega)$: Compute $t = H(e(C_1, Q)^s)$, and check $e(C_1^4, h_\omega)C_3^{2-\omega} \neq C_4$. If the equation holds, output 1, and 0 otherwise.

The Fang et al. SCF-PEKS scheme is secure (in the sense of the non-adaptive security) if the decisional Bilinear Diffie Hellman (DBDH) assumption holds. This is the first SCF-PEKS scheme without random oracles. Note that the Fang et al. SCF-PEKS scheme is not adaptive secure, since there is a trivial attack as follows. Let a challenge ciphertext be $\lambda^* = (C_1^*, C_2^*, C_3^*, C_4^*) = ((g^s, (Yg^{-\omega})^{r/t}, e(g, g)^r, (g, h)^r))$, where $t^* = H(e(C_1, Q)^s)$. In the adaptive-IND-CKA-AT experiment, $A$ chooses $r' \xleftarrow{\$} Z_p$, and computes $C_2^* := (C_2^*)^r$, $C_3^* := (C_3^*)^r$, and $C_4^* := (C_4^*)^r$. Then $\lambda^* := (C_1^*, C_2^*, C_3^*, C_4^*)$ is a valid ciphertext for the keyword $\omega^*_z$, and $\lambda^* \neq \lambda'$. Therefore, $A$ can issue a test query $(\lambda^*, t_{\omega^*_z})$, and outputs 1 if the answer to this query is 1, and 0 otherwise.

We should notice that this attack is positioned in outside of their security models.

Appendix B

Here, we instantiate a non-adaptive SCF-PEKS based on the Gentry (PCS) anonymous IBE [27] and linear encryption presented by Boneh, Boyen, and Shacham [8]. We assume that $e(g, g)$ and $e(g, h)$ are included in public keys to reduce pairing computations.

A non-adaptive SCF-PEKS scheme (the GBBS construction)

SCF-PEKS.KeyGen$_5$(1$^*$): Choose $x, y \in Z_p$ and $u, v, z \in G$ with $u^x = v^y = z$. Output $(pk_S, sk_S) = ((u, v, z), (x, y))$.

SCF-PEKS.KeyGen$_R$(1$^*$): Choose $g, h \xleftarrow{\$} G$ and $\alpha \xleftarrow{\$} Z_p$, compute $g' = g^\alpha$, and output $(pk_R, sk_R) = ((g', h, e(g, g), e(g, h)), \alpha)$.

SCF-PEKS.Trapdoor$(sk_R, \omega)$: For a keyword $\omega \in Z_p$, choose $r_\omega \xleftarrow{\$} Z_p$, compute $h_\omega = (hg^{-r_\omega})^{\frac{1}{x_z}}$, and output $t_\omega = (r_\omega, h_\omega)$.

SCF-PEKS.Enc$(pk_S, pk_R, \omega)$: Choose $R \xleftarrow{\$} G_T$ and $s, r_1, r_2 \xleftarrow{\$} Z_p$. Compute $C_{IBE, 1} = (g^s)^{r_1}, C_{IBE, 2} = (e(g, g)^s, R \cdot e(g, h)^{r_2})$, and $C_{PKE} = (u^{r_1}, u^{r_2}, C_{IBE, 1} \cdot z^{r_1 + r_2})$. Output $\lambda = (C_{IBE, 2}, C_{PKE}, R)$.

SCF-PEKS.Test$(\lambda, sk_S, t_\omega)$: Parse $sk_S = (x, y)$, $t_\omega = (r_\omega, h_\omega)$, $C_{IBE, 2} = (f_1, f_2)$, and $C_{PKE} = (v_1, v_2, v_3)$. Compute $C_{IBE, 1} = v_3/(v_1 \cdot v_2^2)$ and $R' = f_1^{2-\omega} \cdot e(C_{IBE, 1}, h_\omega) \cdot f_2$. Check $R' = R$. If not, then output 0. Otherwise, output 1.
The GBBS construction is secure (in the sense of the non-adaptive security) if the decisional ABDHE assumption and DLIN assumption hold. Note that the GBBS construction is not adaptive secure, since there is a trivial attack as follows. Let \( \lambda^* = (e(g, g)^{s^*}, R^* \cdot e(g, h)^{-s^*}, C_{\text{PKE}}^*, R^*) \) be the challenge ciphertext. Then, choose \( R' \in \mathbb{G}_T \), and compute \( R' \cdot (R^* \cdot e(g, h)^{-s^*}) \) and \( R' \cdot R^* \). Then \( \lambda' = (e(g, g)^{s^*}, R' \cdot R^* \cdot e(g, h)^{-s^*}, C_{\text{PKE}}^*, R' \cdot R^*) \) is a valid ciphertext. Therefore, \( A \) can issue a test query \( (\lambda', t_{\omega^*}) \), and outputs 1 if the answer to this query is 1, and 0 otherwise. To avoid such an attack, TBE and OTS are required in our adaptive SCF-PEKS constructions.