2013

Public key encryption with keyword search secure against keyword guessing attacks without random oracle

Liming Fang
Nanjing University of Aeronautics and Astronautics

Willy Susilo
University of Wollongong, wsusilo@uow.edu.au

Chunpeng Ge
Nanjing University of Aeronautics and Astronautics

Jiandong Wang
Nanjing University of Aeronautics and Astronautics

Publication Details

Fang, L., Susilo, W., Ge, C. & Wang, J. (2013). Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Information Sciences, 238 221-241.

Public key encryption with keyword search secure against keyword guessing attacks without random oracle

Abstract
The notion of public key encryption with keyword search (PEKS) was put forth by Boneh et al. to enable a server to search from a collection of encrypted emails given a "trapdoor" (i.e., an encrypted keyword) provided by the receiver. The nice property in this scheme allows the server to search for a keyword, given the trapdoor. Hence, the verifier can merely use an untrusted server, which makes this notion very practical. Following Boneh et al.'s work, there have been subsequent works that have been proposed to enhance this notion. Two important notions include the so-called keyword guessing attack and secure channel free, proposed by Byun et al. and Baek et al., respectively. The former realizes the fact that in practice, the space of the keywords used is very limited, while the latter considers the removal of secure channel between the receiver and the server to make PEKS practical. Unfortunately, the existing construction of PEKS secure against keyword guessing attack is only secure under the random oracle model, which does not reflect its security in the real world. Furthermore, there is no complete definition that captures secure channel free PEKS schemes that are secure against chosen keyword attack, chosen ciphertext attack, and against keyword guessing attacks, even though these notions seem to be the most practical application of PEKS primitives. In this paper, we make the following contributions. First, we define the strongest model of PEKS which is secure channel free and secure against chosen keyword attack, chosen ciphertext attack, and keyword guessing attack. In particular, we present two important security notions namely IND-SCF-CKCA and IND-KGA. The former is to capture an inside adversary, while the latter is to capture an outside adversary. Intuitively, it should be clear that IND-SCF-CKCA captures a more stringent attack compared to IND-KGA. Second, we present a secure channel free PEKS scheme secure without random oracle under the well known assumptions, namely DLP, DBDH, SXDH and truncated q-ABDHE assumption. Our contributions fill the gap in the literature and hence, making the notion of PEKS

Keywords
without, against, random, oracle, secure, encryption, key, public, guessing, search, keyword, attacks

Disciplines
Engineering | Science and Technology Studies

Publication Details
Fang, L., Susilo, W., Ge, C. & Wang, J. (2013). Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Information Sciences, 238 221-241.

Public Key Encryption with Keyword Search Secure Against Keyword Guessing Attacks without Random Oracle

Liming Fang[1] Willy Susilo[2] * Chunpeng Ge[1] Jiandong Wang[1]

[1] *College of Computer Science and Technology Nanjing University of Aeronautics and Astronautics, 29 Yudao Street, Nanjing, China*

[2] *Centre for Computer and Information Security Research (CCISR), School of Computer Science and Software Engineering, University of Wollongong, Northfields Avenue, NSW 2522, Australia*

Abstract

The notion of public key encryption with keyword search (PEKS) was put forth by Boneh et al. to enable a server to search from a collection of encrypted emails given a "trapdoor" (*i.e.,* an encrypted keyword) provided by the receiver. The nice property in this scheme allows the server to search for a keyword, given the trapdoor. Hence, the verifier can merely use an untrusted server, which makes this notion very practical. Following Boneh et al.'s work, there have been subsequent works that have been proposed to enhance this notion. Two important notions include the so-called *keyword guessing attack* and *secure channel free*, proposed by Byun et al. and Baek et al., respectively. The former realizes the fact that in practice, the space of the keywords used is very limited, while the latter considers the removal of secure channel between the receiver and the server to make PEKS practical. Unfortunately, the existing construction of PEKS secure against keyword guessing attack is only secure under the random oracle model, which does not reflect its security in the real world. Furthermore, there is no complete definition that captures secure channel free PEKS schemes that are secure against chosen keyword attack, chosen ciphertext attack, and against keyword guessing attacks, even though these notions seem to be the most practical application of PEKS primitives. In this paper, we make the following contributions. First, we define the strongest model of PEKS which is secure channel free and secure against chosen keyword attack, chosen ciphertext attack, and keyword guessing attack. In particular, we present two important security notions namely IND-SCF-CKCA and IND-KGA. The former is to capture an inside adversary, while the latter is to capture an outside adversary. Intuitively, it should be clear that IND-SCF-CKCA captures a more stringent attack compared to IND-KGA. Second, we present a secure channel free PEKS scheme secure without random oracle under the well known assumptions, namely DLP, DBDH, SXDH and truncated q-ABDHE assumption. Our contributions fill the gap in the literature

and hence, making the notion of PEKS very practical. We shall highlight that our scheme is IND-SCF-CKCA secure.

Key words: public key encryption with keyword search, keyword guessing attack, without random oracle

1 Introduction

Boneh et al. [4] put forward the notion of public key encryption scheme with keyword search (PEKS scheme), which has a very practical application in an encrypted email system. The basic idea is as follows. Bob sends a ciphertext $\widetilde{C}$, $\widetilde{C} = (C_{PKE} \parallel C_{PEKS}) = (PKE(pk_A, m) \parallel PEKS(pk_A, w))$, to Alice where pk_A is Alice's public key, C_{PKE} is an encrypted version of Bob's message under pk_A and w is the keyword that Bob wants to attach to the email (for example "urgent"). Alice can provide the server with a certain trapdoor T_w (which is a trapdoor constructed by Alice on a keyword w) through a secure channel that enables the server to test whether the encrypted keyword associated with the message (C_{PEKS}) is equal to the keyword w selected by Alice. Given $PEKS(pk_A, w')$ and T_w, the server can test whether $w \stackrel{?}{=} w'$. If $w = w'$, then the server learns nothing more about w'. In short, PEKS provides a mechanism that allows Alice to have the email server extract emails that contain a particular keyword by providing a trapdoor corresponding to the keyword, while the email server and other parties other than Alice will not learn anything else about the email. The construction of Boneh et al.'s PEKS makes use of the construction of Identity-based encryption (IBE) in a very clever way. Following Boneh et al.'s pioneering work [4], Waters et al. [34] demonstrated that the PEKS scheme based on the bilinear pairing could be applied to build encrypted and searchable audit logs. Furthermore, Golle et al. [15], Park et al. [25] and Zhang et al. [36] proposed schemes that allowed conjunctive keyword queries on encrypted data. Subsequently, Boneh and Waters [5] extended PEKS to support conjunctive, subset, and range comparisons over the keywords. Moreover, the subsequent papers [3,12,19,37] investigated the secure combination of public key encryption with keyword search (PEKS) with public key encryption (PKE). Rhee et al. [28] presented two generic transformations to construct a designated tester public-key encryption with keyword search scheme using two identity-based encryption schemes. Recently, Shao

* Corresponding author. Tel: +61-2-4221-5535; Fax: +61-2-4221-4170
Email addresses: fangliming@nuaa.edu.cn (Liming Fang[1]), wsusilo@uow.edu.au (Willy Susilo[2]), gecp@nuaa.edu.cn (Chunpeng Ge[1]), aics@nuaa.edu.cn (Jiandong Wang[1]).

et al. [31] presented the notion of proxy re-encryption with keyword search scheme that combined the public key encryption with keyword search (PEKS) with proxy re-encryption (PRE).

Baek et al. [2] noticed that Boneh et al.'s [4] scheme required a secure channel between the receiver, Alice, and the email server. This makes PEKS scheme impractical, since the original idea of PEKS is to allow Alice to selectively download her emails that contain a particular keyword while she is away. Requiring a secure channel between the receiver and the email server means that the receiver cannot use a regular "untrusted" channel, such as 3G or a public WiFi hot-spot, or at least requiring a secure SSL connection which is quite expensive (and impractical considering the verifier may just use a PDA or an iPhone to download the emails). Subsequently, Baek et al. [2] proposed the notion of PEKS scheme without requiring a secure channel, which is referred to as a secure channel-free PEKS (SCF-PEKS), or sometimes also referred to as the PEKS scheme with a designated tester [27]. The construction provided by Baek et al. relies on the random oracle model, which does not really reflect its security in the real world. In 2007, Gu et al. [14] proposed a more efficient SCF-PEKS scheme in the random oracle model. Recently Fang et al. [10] proposed an SCF-PEKS scheme without random oracle. In SCF-PEKS, only the server (a designated tester) chosen by the receiver is able to perform a test to check the relationship between a ciphertext and a trapdoor. Further, Rhee et al. [27] enhanced Baek et al.'s security model [2] for SCF-PEKS in which an attacker was also allowed to obtain the relationship between non-challenge ciphertexts and a trapdoor. They presented a SCF-PEKS scheme secure in the enhanced security model in the random oracle model.

In practice, everyone will use well-known keywords (with low entropy), such as "urgent", to be attached in the encrypted emails (otherwise the emails cannot be found if the keywords are very unusual). This observation raises the possibility of an important attack to PEKS, namely "keyword guessing attacks", where a malicious attacker can successfully guess some candidates of the keywords and verify the accuracy of this guess in an off-line manner. By performing this off-line keyword guessing attack, the malicious attacker can obtain relevant information from the encrypted emails, and hence, the keyword. This observation was firstly made by Byun et al. [6] who observed that Merriam-Webseter's collegiate dictionary contained only 225,000 keyword definitions. Furthermore, Byun et al. also pointed out that Boneh et al.'s scheme [4] was susceptible to keyword guessing attack. If keyword guessing attack can be launched successfully on a PEKS scheme, the attacker can learn which keyword is used by the sender and the receiver. Thus, the attacker breaks the security of the PEKS scheme. Inspired by the work in Byun et al.[6], Yau et al. [35] presented an off-line keyword guessing attack on the SCF-PEKS [2] and PKE/PEKS [3] schemes.

Two Types of Attackers in SCF-PEKS

We note that there are essentially two types of attackers: (1) the server; and (2) neither the server nor the receiver (*i.e.,* an outsider). As observed by Jeong et al. [20], in the case when the attacker is the server, SCF-PEKS consistency implies insecurity against keyword guessing attacks. To highlight this observation, let us consider the work presented in [29]. In their work, an SCF-PEKS scheme secure against keyword attacks in the random oracle model is presented. If the attacker under consideration is the server, the server knows α, which is the server's private key. Henceforth, the server can perform the test: $dTest(C, T_w, \alpha)$. The keyword guessing attack is performed by the server as follows.

- Step 1: The server captures a valid trapdoor T_w and his main goal is to find out the keyword w from the trapdoor T_w.
- Step 2: The server guesses an appropriate keyword w', and computes a PEKS ciphertext C under keyword w'.
- Step 3: The server tests if $dTest(C, T_w, \alpha) \stackrel{?}{=} 1$. If the equality holds, then the guessed keyword w' is valid. Otherwise, go to Step 2.

Based on this observation, it is not possible to construct an SCF-PEKS scheme secure against keyword guessing attacks, where the attacker is the server. Thereafter, in this work (and also the work by [29]), we do not consider this type of attack. Rather, we focus on analyzing the scheme based on the possible attacks by an outsider (which is neither the server nor the receiver)).

For clarity, we consider an attacker $\mathcal{A}$, neither the server nor the receiver, who conducts off-line keyword guessing attacks on Baek et al.'s SCF-PEKS scheme. For brevity, we will not review Baek et al.'s scheme in this section, but we refer the readers to [2] for more detailed account. As in [29,35], an attacker $\mathcal{A}$, who is neither the server nor the receiver, can perform an off-line keyword attack as follows:

- Step 1: $\mathcal{A}$ first captures a valid trapdoor $T_w = yH_1(w)$.
- Step 2: $\mathcal{A}$ guesses an appropriate keyword w', and computes $H_1(w')$.
- Step 3: $\mathcal{A}$ takes the receiver's public key Y and the hash of the guessed keyword $H_1(w')$, and checks if $e(Y, H_1(w')) = e(P, T_w)$. If so, the guessed keyword w' is a valid keyword. Otherwise, go to Step 2.

The equation holds for $w' = w$, *i.e.*,

$$e(Y, H_1(w')) = e(yP, H_1(w)) = e(P, yH_1(w)) = e(P, T_w).$$

We note that in the keyword guessing attack, we do not need to compute the PEKS ciphertext. We also note that this attack is plausible, since the purpose of SCF-PEKS scheme is to allow the receiver to send a trapdoor via a public channel, and hence, an adversary can also obtain the trapdoor and perform

the attack as outlined above.

Byun et al. [6] also left an open problem on how to construct PEKS schemes secure against keyword guessing attacks. This problem was answered by Rhee et al. [29] by constructing a new secure SCF-PEKS scheme against keyword guessing attacks in the random oracle model. Nevertheless, its security in the real life is questionable, since the random oracle model is required in this scheme.

1.1 Security Vulnerability in Existing Concrete Schemes

Keyword Guessing Attack
We consider two existing schemes namely Baek et al. [2] and Fang et al. [10] as follows.

- (*Baek et al.[2].*) It is clear that Baek et al.'s scheme [2] is vulnerable to keyword guessing attacks as outlined earlier.
- (*Fang et al. [10].*) An attacker $\mathcal{A}$, who is neither the server nor the receiver, can perform an off-line keyword attack as follows:
  · Step 1: $\mathcal{A}$ first captures a valid trapdoor $T_w = (d_w, s_w)$ where $d_w = (hg^{-s_w})^{1/(y-w)}$.
  · Step 2: $\mathcal{A}$ guesses an appropriate keyword w'.
  · Step 3: $\mathcal{A}$ takes the receiver's public key Y and the guessed keyword w', and checks if $e(d_w, Yg^{-w'}) = e(g, hg^{-s_w})$. If so, the guessed keyword w' is a valid keyword. Otherwise, go to Step 2.
  The equation holds for $w' = w$, *i.e.*,

$$e(d_w, Yg^{-w'}) = e((hg^{-s_w})^{1/(y-w)}, g^{y-w}) = e(g, hg^{-s_w}).$$

Chosen ciphertext Attack
A malicious receiver can generate the trapdoor T_{w_0} corresponding to a keyword w_0 using his secret key. Then, the malicious receiver can obtain the relation between the modified chanllenge ciphertext CT' under keyword w_b where $w_b \in \{w_0, w_1\}$ and the trapdoor T_{w_0} through interacting with the email server in real environment.

- (*Fang et al. [10].*) The malicious receiver $\mathcal{A}$ can modify the ciphertext $C^* = (C_1^*, C_2^*, C_3^*, C_4^*) = (g^s, (Yg^{-w_b})^{r/t}, e(g,g)^r, e(g,h)^r)$ to a new valid ciphertext $C' = (C_1^*, (C_2^*)^{r'}, (C_3^*)^{r'}, (C_4^*)^{r'}) = (g^s, (Yg^{-w_b})^{rr'/t}, e(g,g)^{rr'}, e(g,h)^{rr'})$ without knowing the keyword w_b. Then, the malicious receiver can compute the trapdoor $T_{w_0} = (d_{w_0}, s_{w_0})$ since he knows the private key of receiver. The malicious receiver can obtain the relation between the modified challenge ciphertext C' and the trapdoor T_{w_0} through interacting with the email server in real environment.

- (*Rhee et al. [30].*) In [30], firstly let us observe that the ciphertext $C^* = (A^*, B^*) = (pk_{R,1}^r, H_2(e(pk_{S,1}, H_1(w_b)^r)))$ under receiver R where $pk_{R,1} = g^\beta$ is known. Let us assume that adversary $\mathcal{A}$ (i.e., a malicious receiver R) collude with another receiver R'. Since the adversary $\mathcal{A}$ (including the malicious receiver) knows the private key $sk_{R,1} = \beta$ of receiver R, then, with the help of receiver R' , $\mathcal{A}$ can modify the ciphertext to a new valid ciphertext $C' = ((A^*)^{\beta'/\beta}, B^*) = (pk_{R',1}^r, H_2(e(pk_{S,1}, H_1(w_b)^r)))$ under receiver R' where $pk_{R',1} = g^{\beta'}$ without knowing the keyword w_b. This can be done by receiver R' by computing $A' = (A^*)^{\beta'}$ and sending to $\mathcal{A}$, and then, $\mathcal{A}$ computes $(A')^{1/\beta}$(i.e. $(A^*)^{\beta'/\beta}$). Then, the adversary $\mathcal{A}$ can compute the trapdoor $T_{w_0} = (T_1, T_2) = (g^{r'}, H_1(w_0)^{1/\beta'} \cdot H(pk_{S,1}^{r'}))$ with the help of receiver R'. We note that actually the adversary $\mathcal{A}$ can easily obtain the trapdoor T_{w_0} since it is disclosed in the public channel. The adversary $\mathcal{A}$ can obtain the relation between the modified challenge ciphertext C' under receiver R' and the trapdoor T_{w_0} of receiver R' through interacting with the email server in real environment. Clearly, the ciphertext in [29] is similar to that [30], and hence, it is also susceptible to the chosen ciphertext attack.
- (*Rhee et al. [27].*) Rhee et al. introduced the test query in their security model to avoid chosen ciphertext attack in [30], but the ciphertext is of the same form as above, and therefore, it also suffers from the chosen ciphertext attack. That means $\mathcal{A}$ also can modify the ciphertext $C^* = (A^*, B^*) = (pk_{R,1}^r, H_2(e(pk_{S,1}, H_1(w_b)^r)))$ under receiver R to create a new valid ciphertext $C' = ((A^*)^{sk_{R',1}/sk_{R,1}}, B^*) = (pk_{R',1}^r, H_2(e(pk_{S,1}, H_1(w_b)^r)))$ under receiver R' without knowing the keyword w_b. Actually, in their security model, they defined a weaker model that "a malicious outside attacker (including receiver) should not be able to distinguish between the ciphertext of two challenge keywords of its choice under the situation that it is allowed to obtain the relation between a ciphertext $C = (A, B)$ and a trapdoor T_w, where $A \neq A^*$ and $B \neq B^*$ (the ciphertext query is selected among only the ciphertext $C = (A, B)$ not containing the same corresponding element of challenged ciphertext $C^* = (A^*, B^*)$ and T_w is a trapdoor for any non-challenge keywords $w \neq w_0, w_1$".

1.2 Our Contributions

Although Rhee et al.'s schemes [29,30] designed against keyword guessing attacks PEKS is elegant, there remain some important issues to consider:

- (*Random Oracle Model.*) The schemes presented in [29,30] are only proven secure in the random oracle model. Unfortunately, a proof in the random oracle model can only serve as a heuristic argument and admittedly using quite contrived constructions, it has been shown to possibly lead to insecure

schemes when the random oracles are implemented in the standard model [7]. Therefore, it is desirable to construct a secure scheme that does not depend on the random oracle model.

- (*Incomplete Security Definition.*) The solution provided by Rhee et al.[29] lacks of the formal security definition to capture the keyword guessing attacks. Reference Rhee et al. [30] introduced the concept of "trapdoor indistinguishability" and showed that trapdoor indistinguishability is a sufficient condition for thwarting keyword-guessing attacks. The drawback of the security model in [30] is that there is no Test query. We add a stronger Test query, which only restriction if $\langle C, w\rangle = \langle C^*, w_0\rangle$ or $\langle C, w\rangle = \langle C^*, w_1\rangle$.

Based on the above motivations, in this paper, we make the following contributions. First, we provide a complete model, which means that we provide the definition of indistinguishability of secure channel free PEKS against chosen keyword and ciphertext attack (IND-SCF-CKCA) in which we added the test query in IND-SCF-CKA. We also provide the definition of SCF-PEKS secure against keyword guessing attack. Second, we present an efficient SCF-PEKS secure against keyword guessing attacks without requiring the random oracle model. Based on the DLP, DBDH, SXDH and the truncated q-ABDHE assumption, we first prove its indistinguishability of secure channel free PEKS against chosen keyword and ciphertext attack (IND-SCF-CKCA) security without random oracle. Then, we also analyze the computational consistency and security against keyword guessing attacks of our scheme.

In short, this paper fills the gap in the literature by providing the strongest model in SCF-PEKS that is secure against chosen keyword and ciphertext attack (IND-SCF-CKCA) and against keyword guessing attack. Furthermore, our scheme is proven secure under the standard model. To highlight our contribution, we provide the following summary as follows. We selected three schemes from [2,10,29] to represent the state-of-the-art of SCF-PEKS constructions.

	Baek et al. [2]	**Fang et al.** [10]	**Rhee et al.** [29]	**Our scheme**
Without ROM	×	✓	×	✓
Keyword Guessing Attack	×	×	✓	✓
Chosen Ciphertext Security	×	×	×	✓
Assumption	BDH	DBDH, q-ABDHE	BDH, q-BDHI	DBDH, SXDH q-ABDHE

Table 1. Comparison Among Various SCF-PEKS Schemes

Paper Organization

The rest of this paper is organized as follows. In the next section, we will present some definitions and notations that will be used throughout this paper. In Section 3, we present the formal definition of SCF-PEKS secure against keyword guessing attack. In Section 4, we present our new and efficient scheme

and analyze its security. In Section 5, we give a performance comparison of our scheme with the existing schemes in the literature. In Section 6, for completeness, we provide an application where SCF-PEKS is applicable in practice. Finally, Section 7 concludes the paper.

2 Definitions

In this section, we first review the complexity assumption required in our schemes, and then provide the definition and security of a public key encryption with keyword search scheme.

2.1 Negligible Function

A function $\epsilon(n) : N \rightarrow R$ is negligible in n if $1/\epsilon(n)$ is a non-polynomially-bounded quantity in n.

2.2 Bilinear Maps

Let $BMsetup(\lambda)$ be an algorithm that, on input λ, outputs the parameters for a bilinear mapping as $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p, and g be a generator of $\mathbb{G}_1$ and $\widetilde{g}$ be a generator of $\mathbb{G}_2$. (By $\mathbb{G}_1^*$ and $\mathbb{Z}_p^*$, we denote $\mathbb{G}_1 \backslash \{1\}$ where 1 is the identity element of $\mathbb{G}_1$, and $\mathbb{Z}_p \backslash \{0\}$, respectively). We say $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is a bilinear map, if the following conditions hold.

- $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$ for all $a, b \in \mathbb{Z}_p$.
- $e(g, \widetilde{g}) \neq 1$.
- There is an efficient algorithm to compute $e(g_1, \widetilde{g}_2)$ for all $g_1 \in \mathbb{G}_1$ and $\widetilde{g}_2 \in \mathbb{G}_2$.

2.3 The SXDH Assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map, and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the parameters for a bilinear mapping. We define the advantage function $Adv^{SXDH}_{\mathbb{G}_1, \mathcal{B}}(\lambda)$ of an adversary $\mathcal{B}$ as

$$|Pr[\mathcal{B}(g, g^a, g^b, g^{ab}) = 1] - Pr[\mathcal{B}(g, g^a, g^b, g^r) = 1]|$$

where $a, b, r \in \mathbb{Z}_p$ are randomly chosen. The advantage function $Adv^{SXDH}_{\mathbb{G}_2,\mathcal{B}}(\lambda)$ of an adversary $\mathcal{B}$ is defined as

$$|Pr[\mathcal{B}(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^{ab}) = 1] - Pr[\mathcal{B}(\widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^r) = 1]|$$

where $a, b, r \in \mathbb{Z}_p$ are randomly chosen. We say that the symmetric external Diffie-Hellman assumption [16,17] holds if both $Adv^{SXDH}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ and $Adv^{SXDH}_{\mathbb{G}_2,\mathcal{B}}(\lambda)$ are negligible for all PPT $\mathcal{B}$.

2.4 Discrete Logarithm Problem (DLP) [24]

Let $\mathbb{G}$ be multiplicative cyclic groups of prime order p, and g be a generator of $\mathbb{G}$. We define the advantage function $Adv^{DLP}_{\mathcal{B}}(\lambda)$ of an adversary $\mathcal{B}$ as

$$Pr[\mathcal{B}(g, g^a) = a]$$

where $a \in \mathbb{Z}_p$ is randomly chosen. We say that the discrete logarithm problem(DLP) assumption holds if $Adv^{DLP}_{\mathcal{B}}(\lambda)$ is negligible for all PPT $\mathcal{B}$.

2.5 The DBDH Assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map, and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the parameters for a bilinear mapping. We define the advantage function $Adv^{DBDH}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ of an adversary $\mathcal{B}$ as

$$|Pr[\mathcal{B}(g, g^a, g^b, g^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c, e(g, \widetilde{g})^{abc}) = 1]$$

$$-Pr[\mathcal{B}(g, g^a, g^b, g^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c, e(g, \widetilde{g})^r) = 1]|$$

where $a, b, c, r \in \mathbb{Z}_p$ are randomly chosen. We say that the decisional bilinear Diffie Hellman assumption holds if $Adv^{DBDH}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ is negligible for all PPT $\mathcal{B}$.

2.6 The Truncated (Decisional) q-ABDHE Assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the parameters for a bilinear mapping. We define the advantage function $Adv^{q-ABDHE}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ of an adversary $\mathcal{B}$ as

$$|Pr[\mathcal{B}(g, g^x, g^{x^2}, \cdots, g^{x^q}, \widetilde{g}, \widetilde{g}^x, \widetilde{g}^{x^2}, \cdots, \widetilde{g}^{x^q}, \widetilde{g}^z, \widetilde{g}^{zx^{q+2}}, e(g, \widetilde{g})^{zx^{q+1}}) = 1]$$

$$-Pr[\mathcal{B}(g, g^x, g^{x^2}, \cdots, g^{x^q}, \widetilde{g}, \widetilde{g}^x, \widetilde{g}^{x^2}, \cdots, \widetilde{g}^{x^q}, \widetilde{g}^z, \widetilde{g}^{zx^{q+2}}, e(g, \widetilde{g})^r) = 1]|$$

where $x, z, r \in \mathbb{Z}_p$ are randomly chosen. We say that the truncated decisional augmented bilinear Diffie-Hellman exponent (q-ABDHE) assumption [13] holds if $Adv^{q-ABDHE}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ is negligible for all PPT $\mathcal{B}$.

2.7 One-Time Signatures

A one-time signature [8] comprises a triple of algorithms $Sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ such that, on input of a security parameter λ, $\mathcal{G}$ generates a one-time key pair (ssk, svk) while, for any message M, $\mathcal{V}(svk, \sigma, M)$ outputs 1 whenever $\sigma = \mathcal{S}(ssk, M)$ and 0, otherwise. We need strongly unforgeable one-time signatures, which means that no PPT adversary can create a new signature for a previously signed message.

$Sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature if the probability

$$Adv^{OTS} = Pr[(ssk, svk) \leftarrow \mathcal{G}(\lambda); (M, St) \leftarrow F(svk); \sigma \leftarrow \mathcal{S}(ssk, M);$$

$$(M', \sigma') \leftarrow F(M, \sigma, svk, St) : \mathcal{V}(svk, \sigma', M') = 1 \wedge (M', \sigma') \neq (M, \sigma)]$$

where St denotes the state information maintained by F between stages, is negligible for any PPT forger F.

3 Formal Definition of SCF-PEKS Secure Against Keyword Guessing Attacks

3.1 Definition of SCF-PEKS

In the following, we will provide the definition of a SCF-PEKS scheme and the game-based security definition model.

Definition 1 (SCF-PEKS) *A secure channel free public key encryption with keyword search scheme comprises the following algorithms:*

- $GlobalSetup(\lambda)$: *Takes a security parameter λ, generates a global parameter $\mathcal{GP}$.*
- $KeyGen_{Server}(\mathcal{GP})$: *Takes as input the global parameters $\mathcal{GP}$. Outputs the public/secret pair $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ of server $\mathcal{S}$.*
- $KeyGen_{Receiver}(\mathcal{GP})$: *Takes as input $\mathcal{GP}$, generates public/secret pair $(pk_{\mathcal{R}}$, $sk_{\mathcal{R}})$ of receiver $\mathcal{R}$.*
- $PEKS(\mathcal{GP}, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w)$: *Takes as input $\mathcal{GP}$, a receiver's public key $pk_{\mathcal{R}}$, a server's public key $pk_{\mathcal{S}}$, and a keyword w. Outputs a PEKS ciphertext C under w.*

- $Trapdoor(\mathcal{GP}, pk_{\mathcal{S}}, sk_{\mathcal{R}}, w)$: *Takes as input* $\mathcal{GP}$*, a server's public key* $pk_{\mathcal{S}}$*, a receiver's secret key* $sk_{\mathcal{R}}$ *and a keyword* w*. Generates a Trapdoor* T_w*.*
- $Test(\mathcal{GP}, T_w, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C)$: *Takes as input a global parameter* $\mathcal{GP}$*, a Trapdoor* T_w*, a server's secret key* $sk_{\mathcal{S}}$ *and a PEKS ciphertext* $C = PEKS(\mathcal{GP}, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w')$*. Outputs a symbol "Correct" if* $w = w'$ *and "Incorrect" otherwise.*

We define the notion of consistency in a SCF-PEKS scheme, which is similar to the notion of consistency in a PEKS scheme from [1].

Definition 2 (Consistency) *Suppose there exists an adversary* $\mathcal{A}$ *that wants to make consistency fail. The consistency is formally defined as follows:*

$\underline{Exp_{\mathcal{A}}^{cons}(\lambda)}$

$(pk_{\mathcal{R}}, sk_{\mathcal{R}}) \leftarrow KeyGen_{Receiver}(\lambda)$; $(pk_{\mathcal{S}}, sk_{\mathcal{S}}) \leftarrow KeyGen_{Server}(\lambda)$

$(w, w') \leftarrow \mathcal{A}(pk_{\mathcal{R}}, pk_{\mathcal{S}})$

$C \leftarrow PEKS(\mathcal{GP}, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w)$; $T_{w'} \leftarrow Trapdoor(\mathcal{GP}, pk_{\mathcal{S}}, sk_{\mathcal{R}}, w')$

if $w \neq w'$ *and* $Test(\mathcal{GP}, T_{w'}, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C) =$ *"Correct", then return* 1*, else return* 0*.*

We define the advantage of $\mathcal{A}$ *as:*

$$Adv_{\mathcal{A}}^{cons}(\lambda) = Pr[Exp_{\mathcal{A}}^{cons}(\lambda) = 1]$$

The scheme is said to be computationally consistent *if it is negligible for probabilistic polynomial adversary* $\mathcal{A}$ *to win the above experiment.*

In the following, we provide the game-based security definition of SCF-PEKS, which we call indistinguishability of secure channel free PEKS against chosen keyword and ciphertext attack (IND-SCF-CKCA). In other words, IND-SCF-CKCA guarantees that the server that has not obtained the Trapdoors for given keywords cannot tell which PEKS ciphertext encrypts which keyword, and the outside attacker that has not obtained the server's private key cannot make any decisions about the PEKS ciphertexts even though the attacker gets all the trapdoors for the keywords that it holds. (That is, the attacker can see all the trapdoors including the challenges that are sent via a public channel.) Note that the attack model for these two types of attackers is described as $Game_{Server}$ and $Game_{Receiver}$, respectively, in the following definition.

Definition 3 (IND-SCF-CKCA game) *Let* λ *be the security parameter,* $\mathcal{A}$ *be the adversary,* $\mathcal{B}$ *be the challenger, and* KS_w *be the keyword space. We consider the following two games.*

$\underline{Game_{Server}}$: $\mathcal{A}$ is assumed to be a server.

(1) Setup: The global parameter generation algorithm $GlobalSetup(\lambda)$, the two key generation algorithms $KeyGen_{Server}(\mathcal{GP})$ and $KeyGen_{Receiver}(\mathcal{GP})$ are executed. A global parameter $\mathcal{GP}$, private and public key pairs of the receiver and the server, are denoted as $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$and $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ respectively. Then, $\mathcal{B}$ sends $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ and $pk_{\mathcal{R}}$ to $\mathcal{A}$.

(2) Query phase 1. $\mathcal{A}$ makes the queries:

- *Trapdoor query $\langle w \rangle$: $\mathcal{A}$ can adaptively asks $\mathcal{B}$ for the trapdoor query T_w for any keyword $w \in KS_w$ of his choice. $\mathcal{B}$ responds the trapdoor $T_w = Trapdoor(\mathcal{GP}, pk_{\mathcal{S}}, sk_{\mathcal{R}}, w)$ to $\mathcal{A}$.*
- *Test query $\langle C, w \rangle$: $\mathcal{A}$ can adaptively asks $\mathcal{B}$ for the Test query for any keyword w and any PEKS ciphertext of his choice. $\mathcal{B}$ first makes a trapdoor query on $\langle w \rangle$ to get trapdoor T_w and responds the result $Test(\mathcal{GP}, T_w, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C)$ to $\mathcal{A}$.*

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a challenge keyword pair (w_0, w_1). (Notice that none of w_0 nor w_1 has been queried for obtaining a corresponding trapdoor in Phase 1). Upon receiving this, $\mathcal{B}$ responds by choosing a random $\gamma \in \{0, 1\}$, and creates a challenge PEKS ciphertext $C^* = PEKS(\mathcal{GP}, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w_\gamma)$ and sends it to $\mathcal{A}$.

(4) Query phase 2. $\mathcal{A}$ issues a number of trapdoor and test queries as in Phase 1. The restriction here is that w_0 and w_1 are not allowed to be queried as trapdoor queries and $\langle C, w \rangle$ are not allowed to be queried as test queries if $\langle C, w \rangle = \langle C^*, w_0 \rangle$ or $\langle C, w \rangle = \langle C^*, w_1 \rangle$.

(5) Guess. $\mathcal{A}$ outputs the guess γ'. The adversary wins if $\gamma' = \gamma$.

We define $\mathcal{A}$'s advantage in $Game_{Server}$ by $Adv_{\mathcal{A}}^{Game_{Server}}(\lambda) = |Pr[\gamma' = \gamma] - 1/2|$.

$\underline{Game_{Receiver}}$: $\mathcal{A}$ is assumed to be an outside attacker (including the receiver).

(1) Setup: The global parameter generation algorithm $GlobalSetup(\lambda)$, the two key generation algorithms $KeyGen_{Server}(\mathcal{GP})$ and $KeyGen_{Receiver}(\mathcal{GP})$ are executed. A global parameter $\mathcal{GP}$, private and public key pairs of the receiver and the server, are denoted as $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$and $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$, respectively. Then, $\mathcal{B}$ sends $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$ and $pk_{\mathcal{S}}$ to $\mathcal{A}$.

(2) Query phase 1. $\mathcal{A}$ makes the queries:

- *Test query $\langle C, w \rangle$: $\mathcal{A}$ can adaptively asks $\mathcal{B}$ for the Test query for any keyword w and any PEKS ciphertext of his choice. $\mathcal{B}$ first makes a trapdoor query on $\langle w \rangle$ to get trapdoor T_w and responds the result $Test(\mathcal{GP}, T_w, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C)$ to $\mathcal{A}$.*

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a challenge keyword pair (w_0, w_1). Upon receiving this, $\mathcal{B}$ responds by choosing a random $\gamma \in \{0, 1\}$, and creates a challenge PEKS ciphertext $C^* = PEKS(\mathcal{GP}, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w_\gamma)$ and sends it to $\mathcal{A}$.

(4) Query phase 2. $\mathcal{A}$ issues a number of trapdoor and test queries as in Phase 1. The restriction here is that $\langle C, w\rangle$ are not allowed to be queried as test queries if $\langle C, w\rangle = \langle C^*, w_0\rangle$ or $\langle C, w\rangle = \langle C^*, w_1\rangle$. Differently from $Game_{Server}$ that w_0 and w_1 are allowed to be queried as trapdoor queries.
(5) Guess. $\mathcal{A}$ outputs the guess γ'. The adversary wins if $\gamma' = \gamma$.

We define $\mathcal{A}$'s advantage in $Game_{Receiver}$ by $Adv_{\mathcal{A}}^{Game_{Receiver}}(\lambda) = |Pr[\gamma' = \gamma] - 1/2|$. The PEKS scheme is said to be IND-SCF-CKCA secure if $Adv_{\mathcal{A}}^{Game_i}(\lambda)$, where i is either Server or Receiver, is negligible.

3.2 SCF-PEKS Secure Against Keyword Guessing Attacks

In the following, we define the notion of indistinguishability of SCF-PEKS against keyword guessing attack (IND-KGA). Specifically, IND-KGA ensures that an outside adversary (neither the server nor the receiver), that has obtained the trapdoor for challenge keyword cannot observe the relationship between the trapdoor and any keywords.

Definition 4 (IND-KGA game) *Let $\mathcal{A}$ be an outside adversary (neither the server nor the receiver) that makes the KG attack. Let λ be the security parameter, we consider the following game:*

(1) Setup: The global parameter generation algorithm $GlobalSetup(\lambda)$, the two key generation algorithms $KeyGen_{Server}(\mathcal{GP})$ and $KeyGen_{Receiver}(\mathcal{GP})$ are executed. A global parameter $\mathcal{GP}$, private and public key pairs of the receiver and the server are denoted as $(pk_{\mathcal{R}}, sk_{\mathcal{R}})$and $(pk_{\mathcal{S}}, sk_{\mathcal{S}})$ respectively. $\mathcal{B}$ sends ($pk_{\mathcal{R}}$, $pk_{\mathcal{S}}$) to $\mathcal{A}$.
(2) Query phase 1. $\mathcal{A}$ makes the queries:
- *Trapdoor query $\langle w\rangle$: $\mathcal{A}$ can adaptively asks $\mathcal{B}$ for the trapdoor query T_w for any keyword $w \in KS_w$ of his choice. $\mathcal{B}$ responds the trapdoor $T_w = Trapdoor(\mathcal{GP}, pk_{\mathcal{S}}, sk_{\mathcal{R}}, w)$ to $\mathcal{A}$.*

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a challenge keyword pair (w_0, w_1). (Notice that none of w_0 nor w_1 has been queried for obtaining a corresponding trapdoor in Phase 1). Upon receiving this, $\mathcal{B}$ responds by choosing a random $\gamma \in \{0, 1\}$, and creates a challenge $T_{w_\gamma} = Trapdoor(\mathcal{GP}, pk_{\mathcal{S}}, sk_{\mathcal{R}}, w_\gamma)$ and sends it to $\mathcal{A}$.
(4) Query phase 2. $\mathcal{A}$ issues a number of trapdoor queries as in Phase 1. The restriction here is that w_0, w_1 are not allowed to be queried as trapdoor queries.
(5) Guess. $\mathcal{A}$ outputs the guess γ'. The adversary wins if $\gamma' = \gamma$.

We define $\mathcal{A}$'s advantage in IND-KGA game by $Adv_{\mathcal{A}}^{IND-KGA}(\lambda) = |Pr[\gamma' = \gamma] - 1/2|$. The PEKS scheme is said to be IND-KGA attack secure if $Adv_{\mathcal{A}}^{IND-KGA}(\lambda)$ is negligible.

Compared with [30], our IND-SCF-CKCA game is stronger than ciphertext security in [30] because the test query is allowed in our game. Further, we also note that our IND-KGA game is same as the trapdoor security in [30].

4 Efficient SCF-PEKS Scheme Secure Against Keyword Guessing Attack

In this section, we will describe our SCF-PEKS scheme which is IND-SCF-CKCA secure. Subsequently, we will also prove the IND-KGA security of our scheme in the standard model.

4.1 Our Scheme

Prior to presenting our scheme, we first describe the intuition behind our construction. We selected Gentry's IBE scheme in [13] as our initial scheme. After replacing the identity in Gentry's IBE scheme in [13] with the keyword, we obtained the ciphertext $C_2 = (\widetilde{Y}\widetilde{g}^{-w})^r, C_3 = e(g,\widetilde{g})^r, C_4 = e(g,\widetilde{Z})^r$ and the trapdoor $T_w = (d_w = g^{(z-s_w)/(y-w)}, s_w)$. Then, we encrypted C_2 using server's public key by $C_1 = g^s, t = H'(e(X,\widetilde{Q})^s), C_2 = (\widetilde{Y}\widetilde{g}^{-w})^{r/t}$. We shall note that the ciphertext is $C_1 = g^s, C_2 = (\widetilde{Y}\widetilde{g}^{-w})^{r/t}, C_3 = e(g,\widetilde{g})^r, C_4 = e(g,\widetilde{Z})^r$, which are the same as the ciphertext from [10]. However, we note that this scheme is not secure against keyword guessing attack since the adversary can know the keyword from the trapdoor. Concretely, an adversary $\mathcal{A}$, who is neither the server nor the receiver, can perform an off-line keyword attack as follows:

- Step 1: $\mathcal{A}$ first captures a valid trapdoor $T_w = (d_w, s_w)$.
- Step 2: $\mathcal{A}$ guesses an appropriate keyword w'.
- Step 3: $\mathcal{A}$ takes the receiver's public key $\widetilde{Y}$ and the guessed keyword w', and checks if $e(d_w, \widetilde{Y}\widetilde{g}^{-w'}) = e(g, \widetilde{Z}\widetilde{g}^{-s_w})$. If so, the guessed keyword w' is a valid keyword. Otherwise, go to Step 2.

The equation holds for $w' = w$, *i.e.*,

$$e(d_w, \widetilde{Y}\widetilde{g}^{-w'}) = e(g^{(z-s_w)/(y-w)}, \widetilde{g}^{y-w}) = e(g, \widetilde{g}^{(z-s_w)}) = e(g, \widetilde{Z}\widetilde{g}^{-s_w}).$$

To ensure the anonymity of the keyword in the trapdoor, let $d_w = (g^{(z-s_w)/(y-w)})^x = X^{(z-s_w)/(y-w)}$. That means the keyword in the exponent is protected by the combination of the server and receiver's private key (x, y, z). However, the adversary $\mathcal{A}$ can also learn the keyword by testing $e(d_w, \widetilde{Y}\widetilde{g}^{-w'}) = e(X, \widetilde{Z}\widetilde{g}^{-s_w})$. Next, we will introduce Waters' hash function. We review the hash function $H : \{0,1\}^n \to \mathbb{G}_1$ used in Waters' identity based encryption schemes [33]. Choose $n+1$ random group elements $e_0, e_1, \cdots, e_n \in \mathbb{Z}_p$, compute $h_i = g^{e_i}$,

and $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ as the public description of the hash function. The algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as the product

$$h(w) = e_0 + \sum_{i=1}^{n} (e_i \cdot w_i), H(w) = h_0 \prod_{i=1}^{n} (h_i^{w_i}) = g^{h(w)}.$$

We can view the part $h(w)$ as the private key, and use this to protect the keyword, and thus we change the trapdoor to $d_w = (X^{(z-s_w)/(y-w)})^{h(w)}$. As a result, the ciphertext is

$$CT = (C_1, C_2, C_3, C_4) = (g^s, (\widetilde{Y}\widetilde{g}^{-w})^{r/t}, e(H(w), \widetilde{g})^r, e(H(w), \widetilde{Z})^r).$$

The drawback of this ciphertext is that the adversary $\mathcal{A}$ can modify the ciphertext CT to produce a new valid ciphertext CT' without knowing the plaintext. Further, a malicious receiver can generate the trapdoor $T_{w'}$ corresponding to a guessed keyword w' using his secret key. Then, the malicious receiver can obtain the relation between the modified challenge ciphertext CT' and the trapdoor $T_{w'}$ through interacting with the email server in real environment. This is why we need a Test query in the security model. To address this issue, we introduce the strongly unforgeable one-time signatures $\sigma = \mathcal{S}(ssk, (C_1, C_2, C_3, C_4, C_5))$ on the pair $(C_1, C_2, C_3, C_4, C_5)$ where $C_5 = (\widetilde{u}^{svk}\widetilde{v})^s$.

We will now present our scheme in detail. Our scheme is defined as follows.

- *GlobalSetup*(λ)*:* Let λ be the security parameter and $\gamma = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the bilinear map parameters. Specify a cryptographically secure pseudorandom number generator (PRNG)[32,26,11] $H' : \mathbb{G}_T \to \mathbb{Z}_p^*$. Generators $\widetilde{u}, \widetilde{v} \in \mathbb{G}_2$ and a strongly unforgeable one-time signature scheme $sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The global parameters are $\mathcal{GP} = (p, g, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \widetilde{u}, \widetilde{v}, sig, H')$.
- $KeyGen_{Server}(\mathcal{GP})$*:* Selects $x \in \mathbb{Z}_p$ uniformly at random and compute $X = g^x$. Selects $\widetilde{Q} \in \mathbb{G}_2^*$ uniformly at random. Outputs $pk_{\mathcal{S}} = (\widetilde{Q}, X)$ and $sk_{\mathcal{S}} = x$ as the server's public and private key, respectively.
- $KeyGen_{Receiver}(\mathcal{GP})$*:* Let the keyword space be $KS_w = \{0,1\}^n$ where $2^n \ll p$. Chooses $n+1$ random group elements $e_0, e_1, \cdots, e_n \in \mathbb{Z}_p$, computes $h_i = g^{e_i}$, and $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ as the public description of the hash function. The algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as the product

$$h(w) = e_0 + \sum_{i=1}^{n} (e_i \cdot w_i), H(w) = h_0 \prod_{i=1}^{n} (h_i^{w_i}) = g^{h(w)}$$

Chooses $y, z \in \mathbb{Z}_p^*$ uniformly at random and computes $\widetilde{Y} = \widetilde{g}^y, \widetilde{Z} = \widetilde{g}^z$.

Outputs $pk_{\mathcal{R}} = (\widetilde{Y}, \widetilde{Z}, h_0, h_1, \cdots, h_n)$ and $sk_{\mathcal{R}} = (y, z, e_0, e_1, \cdots, e_n)$ as the receiver's public and private key, respectively.

- *$PEKS(\mathcal{G}P, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w)$:*

(1) Selects a one-time signature key pair $(ssk, svk) \leftarrow \mathcal{G}(\lambda)$ and sets $C_0 = svk$.
(2) Picks $s, r \in \mathbb{Z}_p^*$ and computes
$C_1 = g^s$, $t = H'(e(X, \widetilde{Q})^s)$, $C_2 = (\widetilde{Y}\widetilde{g}^{-w})^{r/t}$
$C_3 = e(H(w), \widetilde{g})^r$, $C_4 = e(H(w), \widetilde{Z})^r$,$C_5 = (\widetilde{u}^{svk}\widetilde{v})^s$.
(3) Generates a one-time signature $\sigma = \mathcal{S}(ssk, (C_1, C_2, C_3, C_4, C_5))$ on the pair $(C_1, C_2, C_3, C_4, C_5)$.
(4) The PEKS ciphertext is $C = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$. Outputs C.

- *$Trapdoor(\mathcal{G}P, pk_{\mathcal{S}}, sk_{\mathcal{R}}, w)$:* Chooses $s_w \in \mathbb{Z}_p^*$ and computes

$$d_w = X^{h(w)(z-s_w)/(y-w)}$$

Let the trapdoor be $T_w = (d_w, s_w)$. Outputs T_w.

- *$Test(\mathcal{G}P, T_w, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C)$:* Tests if

$$\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1 \tag{1}$$
$$e(C_1, \widetilde{u}^{C_0}\widetilde{v}) = e(g, C_5) \tag{2}$$

Computes $t = H'(e(C_1, \widetilde{Q})^x)$ and checks if

$$e(d_w, C_2^{t/x})C_3^{s_w} = C_4 \tag{3}$$

If all equations (1), (2) and (3) hold, then return "Correct". Otherwise, return "Incorrect".

Correctness. Here we show that a correctly generated PEKS ciphertext can be correctly tested by the server equipped with a correct trapdoor. In the following, let $C = (C_1, C_2, C_3, C_4)$ be a PEKS ciphertext associated with keyword w under the public key $pk_{\mathcal{S}}, pk_{\mathcal{R}}$. Let the trapdoor be $T_w = (d_w, s_w)$. We have

$$\begin{aligned}
t &= H'(e(C_1, \widetilde{Q})^x) \\
&= H'(e(g^s, \widetilde{Q})^x) \\
&= H'(e(g^x, \widetilde{Q})^s) \\
&= H'(e(X, \widetilde{Q})^s). \\
e(d_w, C_2^{t/x})C_3^{s_w} &= e(X^{h(w)(z-s_w)/(y-w)}, ((\widetilde{Y}\widetilde{g}^{-w})^{r/t})^{t/x})(e(H(w), \widetilde{g})^r)^{s_w} \\
&= e(g, \widetilde{g})^{h(w)(z-s_w)r} e(g, \widetilde{g})^{h(w)rs_w} \\
&= e(g, \widetilde{g})^{h(w)zr} \\
&= C_4.
\end{aligned}$$

□

4.2 Consistency of Our SCF-PEKS

In this subsection, we prove the computational consistency of our scheme.

Theorem 1 *Our SCF-PEKS scheme is computationally consistent assuming the discrete logarithm problem (DLP) assumption holds.*

Proof. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, that can attack computational consistent of our scheme. We build a simulator $\mathcal{B}$ that can play a DLP game. Simulator $\mathcal{B}$ inputs a DLP instance (g, g^y) and has to compute y. Let $Adv_{\mathcal{B}}^{DLP}(\lambda)$ be the advantage function that $\mathcal{B}$ solves DLP in $\mathbb{G}_1$. Let (w, w') denote the pair of keywords that $\mathcal{A}$ returns in the consistency experiment, and assume without loss of generality that $w \neq w'$.

Let $s, r \in \mathbb{Z}_p^*$ denote the value chosen at random by $PEKS(\mathcal{GP}, pk_{\mathcal{S}}, pk_{\mathcal{R}}, w)$. $(ssk, svk) \leftarrow \mathcal{G}(\lambda)$ be a one-time signature key pair and let $C_0 = svk$, $C_1 = g^s$, $t = H(e(X, \widetilde{Q})^s)$, $C_2 = (\widetilde{Y}\widetilde{g}^{-w})^{r/t}$, $C_3 = e(H(w), \widetilde{g})^r$, $C_4 = e(H(w), \widetilde{Z})^r$,$C_5 = (\widetilde{u}^{svk}\widetilde{v})^s$.

Let $T_{w'} = (d_{w'}, s_{w'})$ where $d_{w'} = X^{h(w')(z-s_{w'})/(y-w')}$ is the trapdoor of w'

Note that $\mathcal{A}$ wins exactly when $w \neq w'$ and $e(d_{w'}, C_2^{t/x})C_3^{s_{w'}} = C_4$.

$e(d_{w'}, C_2^{t/x})C_3^{s_{w'}} = C_4$

$\Longleftrightarrow e(X^{h(w')(z-s_{w'})/(y-w')}, (\widetilde{Y}\widetilde{g}^{-w})^{r/t\cdot t/x})e(H(w), \widetilde{g})^{rs_{w'}} = e(H(w), \widetilde{Z})^r$

$\Longleftrightarrow e(g^{h(w')(z-s_{w'})/(y-w')}, (\widetilde{g}^{(y-w)r})e(g, \widetilde{g})^{h(w)s_{w'}r} = e(g, \widetilde{g})^{h(w)zr}$

$\Longleftrightarrow e(g^{(h(w')(y-w)/(y-w'))(z-s_{w'})r}, \widetilde{g})e(g, \widetilde{g})^{h(w)rs_{w'}} = e(g, \widetilde{g})^{h(w)zr}$

$\Longleftrightarrow e(g^{((h(w')(y-w)/(y-w'))zr}, \widetilde{g})e(g, \widetilde{g})^{-((h(w')(y-w)/(y-w'))s_{w'}r}e(g, \widetilde{g})^{h(w)rs_{w'}}$
$= e(g, \widetilde{g})^{h(w)zr}$

$\Longleftrightarrow (h(w')(y-w)/(y-w'))zr - (h(w')(y-w)/(y-w'))s_{w'}r + h(w)rs_{w'} = h(w)zr \ mod \ p$

$\Longleftrightarrow (h(w')(y-w)/(y-w') - h(w))zr - (h(w')(y-w)/(y-w') - h(w))s_{w'}r = 0 \ mod \ p$

$\Longleftrightarrow [h(w')(y-w) - h(w)(y-w')]/(y-w')zr - [h(w')(y-w) - h(w)(y-w')]/(y-w')s_{w'}r = 0 \ mod \ p$

$\Longleftrightarrow [h(w')(y-w) - h(w)(y-w')]/(y-w')(z-s_{w'})r = 0 \ mod \ p$

Note that $s_{w'}$ is randomly selected by receiver in $\mathbb{Z}_p^*$. Therefore, for a fixed $z \in \mathbb{Z}_p^*$, $Pr[s_{w'} = z \ mod \ p \ ] = \frac{1}{p-1}$. Further, when $h(w') = h(w) \neq 0 \ mod \ p$, since $h(w')(y-w) - h(w)(y-w') = h(w)(w'-w) \neq 0 \ mod \ p$, thus we have $y = w' \ mod \ p$. When $h(w') \neq h(w) \ mod \ p$, we have $y = \frac{h(w')w - h(w)w'}{h(w') - h(w)} \ mod \ p$ or $y = w' \ mod \ p$, and thus, we can solve the discrete logarithm problem as claimed. Next, we will discuss that $Pr[h(w) = 0] = \frac{2^n}{p}$. Since $e_0, e_1, \cdots, e_n$ are randomly chosen in $\mathbb{Z}_p$, for fixed w, $Pr[h(w) = 0 \ mod \ p \ ] = Pr[\Sigma_{w_i=1} e_i = 0 \ mod \ p \ ] = \frac{1}{p}$. The total number of different w is 2^n, and thus we have $Pr[h(w) = 0 \ mod \ p \ ] = \frac{2^n}{p}$. Actually, our hash function $H(w) = h_0 \prod_{i=1}^{n} (h_i^{w_i})$ is the same as Waters' hash function $H(ID) = h_0 \prod_{i=1}^{n} (h_i^{ID_i})$ in identity based encryption schemes [33,23], since the private key of ID is constructed as $(g_2^{\alpha} H(ID)^r, g^r)$, and if $h(ID) = 0$, it can compute the master key g_2^{α} of the Waters' IBE scheme.

As described above, under the condition $w \neq w'$ and $Test(\mathcal{GP}, T_{w'}, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C)$ = "$Correct$"

$$Adv_{\mathcal{A}}^{cons}(\lambda) = Pr[Exp_{\mathcal{A}}^{cons}(\lambda) = 1] \leq \frac{1}{p-1} + \frac{2^n}{p} + Adv_{\mathcal{B}}^{DLP}(\lambda).$$

□

4.3 Security of Our SCF-PEKS

In this subsection, we will prove the security of our SCF-PEKS scheme without any random oracle. The analysis of $Game_{Server}$ and $Game_{Receiver}$ is as follows.

Theorem 2 *The above scheme is IND-SCF-CKCA secure without random oracle model assuming that H' is a cryptographically secure pseudorandom number generator (PRNG) and that the DBDH problem and q-ABDHE problem are intractable.*

Proof. The proof of this theorem will be provided in the following two lemmas, for clarity. These lemmas represent $Game_{Server}$ and $Game_{Receiver}$, respectively, as defined in Definition 3.

Lemma 1. Let $q \geq q_k + 1$, where q_k is the number of trapdoor queries. Our scheme is semantically secure against a chosen keyword and ciphertext attacks in $Game_{Server}$ without the random oracle model assuming q-ABDHE problem is intractable.

Proof. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, in $Game_{Server}$ that can attack our scheme in the standard model. Let q_k is the number of

trapdoor queries. We build a simulator $\mathcal{B}$ that can play a q-ABDHE game. The simulation proceeds as follows:

We first let the challenger set the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ with an efficient bilinear map e and a generator g of $\mathbb{G}_1$. Simulator $\mathcal{B}$ inputs a q-ABDHE instance

$$(g, g^x, g^{x^2}, \cdots, g^{x^q}, \widetilde{g}, \widetilde{g}^x, \widetilde{g}^{x^2}, \cdots, \widetilde{g}^{x^q}, \widetilde{g}^z, \widetilde{g}^{zx^{q+2}}, T)$$

and has to distinguish $T = e(g, \widetilde{g})^{zx^{q+1}}$ from a random element in $\mathbb{G}_T$.

(1) Setup: Let λ be the security parameter and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the bilinear map parameters. Specify a cryptographically secure pseudorandom number generator (PRNG) $H' : \mathbb{G}_T \to \mathbb{Z}_p^*$. Generators $\widetilde{u}, \widetilde{v} \in \mathbb{G}_2$ and a strongly unforgeable one-time signature scheme $sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The global parameters are $\mathcal{G}P = (p, g, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \widetilde{u}, \widetilde{v}, sig, H')$.

Choose $a \in \mathbb{Z}_p^*$ uniformly at random and computes $X = g^a$. Chooses $\widetilde{Q} \in \mathbb{G}_2^*$ uniformly at random. Output $pk_{\mathcal{S}} = (\widetilde{Q}, X)$ and $sk_{\mathcal{S}} = a$ as the server's public and private key, respectively.

Let the keyword space $KS_w = \{0,1\}^n$. Choose $n+1$ random group elements $e_0, e_1, \cdots, e_n \in \mathbb{Z}_p$, and compute $h_i = g^{e_i}$, and $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ as the public description of the hash function. The algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as the product

$$h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i), H(w) = h_0 \prod_{i=1}^{n}(h_i^{w_i}) = g^{h(w)}$$

Pick a random degree q polynomials $f(X)$, and define $\widetilde{Y} = \widetilde{g}^x, \widetilde{Z} = \widetilde{g}^{f(x)}$. The receiver's public and private keys are $pk_{\mathcal{R}} = (\widetilde{Y}, \widetilde{Z}, h_0, h_1, \cdots, h_n)$, $sk_{\mathcal{R}} = (x, f(x), e_0, e_1, \cdots, e_n)$, respectively (Note that $\mathcal{B}$ cannot know x and cannot compute $f(x)$ as well), and $(pk_{\mathcal{R}}, pk_{\mathcal{S}}, sk_{\mathcal{S}})$ is sent to $\mathcal{A}$.

(2) Query phase 1. $\mathcal{A}$ makes the following queries:
- Trapdoor query $\langle w \rangle$: If $\mathcal{A}$ queries w to the trapdoor generation oracle, then $\mathcal{B}$ sets $s_w = f(w)$, computes $d_w = (g^{(f(x)-f(w))/(x-w)})^{ah(w)}$, and sends the trapdoor $T_w = \{d_w, s_w\}$ to $\mathcal{A}$. When $q \geq q_k + 1$, $s_w = f(w)$ is a random value from $\mathcal{A}$'s view, since $f(X)$ is a random degree q polynomial.
- Test query $\langle C, w \rangle$: $\mathcal{A}$ can adaptively asks $\mathcal{B}$ for the test query for any keyword w and any PEKS ciphertext of his choice. $\mathcal{B}$ first query a trapdoor query on $\langle w \rangle$ to get the trapdoor T_w and then responds by sending the result $Test(\mathcal{G}P, T_w, pk_{\mathcal{S}}, sk_{\mathcal{S}}, C)$ to $\mathcal{A}$.

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a keyword pair (w_0, w_1). $\mathcal{B}$ responds by choosing a random $\gamma \in \{0,1\}$, let $w^* = w_\gamma$, selects a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$, sets $C_0^* = svk^*$

and $\{s_{w^*} = f(w^*)\}$, then $\mathcal{B}$ computes

$$d_{w^*} = (g^{(f(x)-f(w^*))/(x-w^*)})^{ah(w^*)}$$

$\mathcal{B}$ randomly chooses $s^* \in \mathbb{Z}_p^*$ and computes

$$C_1^* = g^{s^*}, t^* = H'(e(X, \widetilde{Q})^{s^*}).$$

$\mathcal{B}$ defines the degree $q+1$ polynomial

$$F^*(X) = (X^{q+2} - (w^*)^{q+2})/(X - w^*) = \sum_{i=0}^{q+1}(F_i^* X^i).$$

$\mathcal{B}$ computes

$$\begin{aligned}
C_2^* &= (\widetilde{g}^{zx^{q+2}}(\widetilde{g}^z)^{-(w^*)^{q+2}})^{1/t^*} \\
C_3^* &= (T^{F_{q+1}^*} e(\prod_{i=0}^{q}(g^{x^i})^{F_i^*}, \widetilde{g}^z))^{h(w^*)} \\
C_4^* &= e(d_{w^*}, (C_2^*)^{t^*/a})(C_3^*)^{s_{w^*}} \\
C_5^* &= (\widetilde{u}^{svk^*}\widetilde{v})^{s^*}
\end{aligned}$$

$\mathcal{B}$ generates a one-time signature $\sigma^* = \mathcal{S}(ssk^*, (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*))$.

$\mathcal{B}$ sends the challenge PEKS ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$ to $\mathcal{A}$.

Let $r^* = zF^*(x)$, if $T = e(g, \widetilde{g})^{zx^{q+1}}$, then

$$\begin{aligned}
C_2^* &= (\widetilde{g}^{zx^{q+2}}(\widetilde{g}^z)^{-(w^*)^{q+2}})^{1/t^*} \\
&= \widetilde{g}^{(x-w^*)(z(x^{q+2}-(w^*)^{q+2})/(x-w^*))1/t^*} \\
&= \widetilde{g}^{(x-w^*)r^*/t^*} = (\widetilde{Y}\widetilde{g}^{-w^*})^{r^*/t^*} \\
C_3^* &= (T^{F_{q+1}^*} e(\prod_{i=0}^{q}(g^{x^i})^{F_i^*}, \widetilde{g}^z))^{h(w^*)} \\
&= e(H(w^*), \widetilde{g})^{r^*} \\
C_4^* &= e(H(w^*), \widetilde{Z})^{r^*}
\end{aligned}$$

(4) Query phase 2. $\mathcal{A}$ continues making queries as in the Query phase 1. The restriction here is that w_0 and w_1 are not allowed to be queried as trapdoor queries and $\langle C, w\rangle$ are not allowed to be queried as test queries if $\langle C, w\rangle = \langle C^*, w_0\rangle$ or $\langle C, w\rangle = \langle C^*, w_1\rangle$.

(5) Guess. $\mathcal{A}$ outputs the guess γ', if $\gamma' = \gamma$, then $\mathcal{B}$ outputs 1 meaning $T = e(g, \widetilde{g})^{zx^{q+1}}$; else $\mathcal{B}$ outputs 0 meaning $T = e(g, \widetilde{g})^r$.

Probability Analysis: If $T = e(g, \widetilde{g})^{zx^{q+1}}$, then the simulation is perfect, and $\mathcal{A}$ will guess the bit γ correctly with probability $1/2 + \epsilon$. Else, T is uniformly

random, and thus (C_2^*, C_3^*) is a uniformly random and independent element. In this case, the inequality

$$C_3^* \neq e(g, (C_2^*)^{t^* h(w^*)})^{1/(x-w^*)}$$

holds with probability $1 - 1/p$. When the inequality holds, the value of

$$C_4^* = e(d_{w^*}, (C_2^*)^{t^*/a})(C_3^*)^{s_{w^*}}$$

$$= e(X^{h(w^*)(f(x)-s_{w^*})/(x-w^*)}, (C_2^*)^{t^*/a})(C_3^*)^{s_{w^*}}$$

$$= e(g^{f(x)/(x-w^*)}, (C_2^*)^{t^* h(w^*)})((C_3^*)/(e(g, (C_2^*)^{t^* h(w^*)})^{1/(x-w^*)}))^{s_{w^*}}$$

is uniformly random and independent from $\mathcal{A}$'s view (except for the value C_4^*), since s_{w^*} is uniformly random (when $q \geq q_k+1$, $s_{w^*} = f(w^*)$ are random values from $\mathcal{A}$'s view) and independent from $\mathcal{A}$'s view (except for the value C_3^*). Thus, C_4^* is uniformly random and independent. Since $s^* \in \mathbb{Z}_p^*$ is randomly chosen, $C_1^* = g^{s^*}$ is uniformly random and independent from (C_2^*, C_3^*, C_4^*) and $(C_1^*, C_2^*, C_3^*, C_4^*)$ can reveal no information regarding the bit γ.

We now summarize the above statements into a bound on the advantage of the adversary in the $Game_{Server}$:

$$Adv_{\mathcal{A}}^{Game_{Server}}(\lambda) \leq Adv_{\mathbb{G}_1,\mathcal{B}}^{q-ABDHE}(\lambda) + 1/p.$$

Lemma 2. Our scheme is semantically secure against a chosen keyword attack in $Game_{Receiver}$ without random oracle model assuming DBDH problem is intractable, $Sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature and H' is a cryptographically secure pseudorandom number generator (PRNG).

Proof. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, in $Game_{Receiver}$ that can attack our scheme in the standard model. We build a simulator $\mathcal{B}$ that can play a DBDH game.

We first let the challenger set the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ with an efficient bilinear map e and a generator g of $\mathbb{G}_1$. $\mathcal{B}$ inputs a DBDH instance $(g, g^a, g^b, g^c, \tilde{g}, \tilde{g}^a, \tilde{g}^b, \tilde{g}^c, T)$, and has to distinguish $T = e(g, \tilde{g})^{abc}$ from a random element in $\mathbb{G}_T$.

Before describing $\mathcal{B}$, we first define an event F_{OTS} and bound its probability to occur. Let $C^* = (svk^*, C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$ denote the challenge ciphertext given to $\mathcal{A}$ in the game. Let F_{OTS} be the event that $\mathcal{A}$ issues a test query for ciphertext $C^* = (svk^*, C_1, C_2, C_3, C_4, C_5, \sigma)$ but $\mathcal{V}(svk^*, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1$. In the "phase 1" stage, $\mathcal{A}$ has simply no information on svk^*. Hence, the probability of a pre-challenge occurrence of F_{OTS} does not exceed $q_k\theta$ if q_k is

the overall number of test oracle queries and θ denotes the maximal probability (which by assumption does not exceed $1/p$) that any one-time verification key svk^* is output by $\mathcal{G}$. In the "phase 2" stage, F_{OTS} clearly gives rise to an algorithm breaking the strong unforgeability of the one-time signature. Therefore, the probability $Pr[F_{OTS}] \leq q_k/p + Adv^{OTS}$, where the second term accounts for the probability of definition one time signature, must be negligible by assumption.

We now proceed with the description of $\mathcal{B}$ that simply halts and outputs a random bit if F_{OTS} occurs. In a preparation phase, $\mathcal{B}$ generates a one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$ and provides $\mathcal{A}$ with public parameters including $\widetilde{u} = (\widetilde{g}^b)^{\alpha_1}$ and $\widetilde{v} = (\widetilde{g}^b)^{-\alpha_1 svk^*}\widetilde{g}^{\alpha_2}$ for random $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$. Throughout the game, $\mathcal{A}$'s environment is simulated as follows.

(1) Setup: Let λ be the security parameter and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the bilinear map parameters. Specify a cryptographically secure pseudorandom number generator (PRNG) $H' : \mathbb{G}_T \to \mathbb{Z}_p^*$. $\mathcal{B}$ generates a strongly unforgeable one-time signature scheme $sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The global parameters are

$$\mathcal{G}P = (p, g, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \widetilde{u}, \widetilde{v}, sig, H').$$

Let $X = g^a$ and $\widetilde{Q} = \widetilde{g}^b$, the server's public key is $pk_{\mathcal{S}} = (\widetilde{Q}, X)$.

Let the keyword space $KS_w = \{0,1\}^n$. $\mathcal{B}$ chooses $n+1$ random group elements $e_0, e_1, \cdots, e_n \in \mathbb{Z}_p$, and computes $h_i = g^{e_i}$, and $h = (h_0, h_1, \cdots, h_n) \in \mathbb{G}_1^{n+1}$ as the public description of the hash function. The algebraic hash function $H : \{0,1\}^n \to \mathbb{G}_1$ is evaluated on a keyword string $w = (w_1, \cdots, w_n) \in \{0,1\}^n$ as the product

$$h(w) = e_0 + \sum_{i=1}^{n}(e_i \cdot w_i), H(w) = h_0 \prod_{i=1}^{n}(h_i^{w_i}) = g^{h(w)}$$

Chooses $y, z \in \mathbb{Z}_p^*$ uniformly at random and computes $\widetilde{Y} = \widetilde{g}^y, \widetilde{Z} = \widetilde{g}^z$.

Outputs $pk_{\mathcal{R}} = (\widetilde{Y}, \widetilde{Z}, h_0, h_1, \cdots, h_n)$ and $sk_{\mathcal{R}} = (y, z, e_0, e_1, \cdots, e_n)$ as the receiver's public and private key respectively, and sends $(pk_{\mathcal{R}}, sk_{\mathcal{R}}, pk_{\mathcal{S}})$ to $\mathcal{A}$.

(2) Query phase 1. $\mathcal{A}$ makes trapdoor and test queries :

- Test query $\langle C, w\rangle$: $\mathcal{A}$ can adaptively ask $\mathcal{B}$ for the test query for any keyword w and any PEKS ciphertext $C = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ of his choice. $\mathcal{B}$ tests if

$$\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1$$

$$e(C_1, \widetilde{u}^{C_0}\widetilde{v}) = e(g, C_5)$$

If the equality holds, then there are two cases:

· If $C_0 = svk \equiv svk^* = C_0^*$, then

$$(C_1, C_2, C_3, C_4, C_5, \sigma) \neq (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$$

$\mathcal{B}$ is faced with an occurrence of F_{OTS} and halts (an occurrence of F_{OTS} in phase 1 and phase 2 are different from that discussed in the preparation phase).

· If $C_0 = svk \neq svk^* = C_0^*$, The validity of the ciphertext ensures that

$$e(C_1, \widetilde{u}^{C_0}\widetilde{v}) = e(g, C_5)$$
$$C_5 = (\widetilde{u}^{svk}\widetilde{v})^s = ((\widetilde{g}^b)^{\alpha_1 svk}(\widetilde{g}^b)^{-\alpha_1 svk^*}\widetilde{g}^{\alpha_2})^s$$
$$= ((\widetilde{g}^{bs})^{\alpha_1(svk-svk^*)}\widetilde{g}^{s\alpha_2})$$

and since $C_1 = g^s$, $\mathcal{B}$ can compute

$$e(g^a, \widetilde{g}^{bs}) = (e(g^a, C_5)/e(C_1^{\alpha_2}, \widetilde{g}^a))^{1/(\alpha_1(svk-svk^*))}.$$

Then, $\mathcal{B}$ can compute

$$t = H'(e(X, \widetilde{Q})^s) = H'(e(g^a, \widetilde{g}^b)^s) = H'(e(g^a, \widetilde{g}^{bs}))$$
$$= H'((e(g^a, C_5)/(e(C_1^{\alpha_2}, \widetilde{g}^a)))^{1/(\alpha_1(svk-svk^*))}))$$

Then $\mathcal{B}$ chooses $s_w \in \mathbb{Z}_p^*$ and computes $d_w = X^{h(w)(z-s_w)/(y-w)}$. $\mathcal{B}$ checks if

$$e(d_w, C_2^{t/x})C_3^{s_w} = C_4$$

If all equations hold, return "Correct", and "Incorrect" otherwise.

(3) Challenge. Once $\mathcal{A}$ decides that phase 1 is over, it outputs a keyword pair (w_0, w_1). $\mathcal{B}$ responds by choosing a random $\gamma \in \{0, 1\}$, let the challenge keyword $w^* = w_\gamma$, sets $C_0^* = svk^*$, $C_1^* = g^c$ and $t^* = H'(T)$, chooses $r^* \in \mathbb{Z}_p^*$, computes $C_2^* = (\widetilde{Y}\widetilde{g}^{-w^*})^{r^*/t^*}$, $C_3^* = e(H(w^*), \widetilde{g})^{r^*}$, $C_4^* = e(H(w^*), \widetilde{Z})^{r^*}$, $C_5^* = (\widetilde{u}^{svk^*}\widetilde{v})^c = ((\widetilde{g}^b)^{\alpha_1 svk^*}(\widetilde{g}^b)^{-\alpha_1 svk^*}\widetilde{g}^{\alpha_2})^c = (\widetilde{g}^c)^{\alpha_2})$.

Generates a one-time signature $\sigma^* = \mathcal{S}(ssk^*, (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*))$

Sends the challenge PEKS ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$ to $\mathcal{A}$.

(4) Query phase 2. $\mathcal{A}$ issues a number of trapdoor and test queries as in phase 1. The restriction here is that $\langle C, w\rangle$ are not allowed to be queried as test queries if $\langle C, w\rangle = \langle C^*, w_0\rangle$ or $\langle C, w\rangle = \langle C^*, w_1\rangle$. Unlike $Game_{Server}$, here w_0 and w_1 are allowed to be queried as trapdoor queries.

(5) Guess. $\mathcal{A}$ outputs the guess γ'. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs 1 meaning $T = e(g, \widetilde{g})^{abc}$; else $\mathcal{B}$ outputs 0 meaning $T = e(g, \widetilde{g})^r$.

Probability Analysis: Suppose there exists a polynomial-time adversary, $\mathcal{A}$, in $Game_{Receiver}$ that can attack our scheme in the standard model with an advantage ε. Now we determine the probability of the simulator $\mathcal{B}$:

If F_{OTS} does not occur, when $T = e(g, \widetilde{g})^{abc}$, then $\mathcal{A}$ must satisfy $|Pr[\gamma' = \gamma] - 1/2| \geq \epsilon$. When T is uniform in $\mathbb{G}_T$, then $t^* = H'(T)$ is uniform in $\mathbb{Z}_p^*$, due to the fact that PRNG is cryptographically secure. r^* is also uniform in $\mathbb{Z}_p^*$ from $\mathcal{A}$'s view, then w^* in $C_2^* = (\widetilde{Y}\widetilde{g}^{-w^*})^{r^*/t^*}$, $C_3^* = e(H(w^*), \widetilde{g})^{r^*}$,

$C_4^* = e(H(w^*), \widetilde{Z})^{r^*}$ is perfectly hidden, then $|Pr[\gamma' = \gamma] = 1/2$. Therefore, when a, b, c are uniform in $\mathbb{Z}_p^*$ and T is uniform in $\mathbb{G}_T$, we have

$$\begin{aligned} &|Pr[\mathcal{B}(g, g^a, g^b, g^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c, e(g, \widetilde{g})^{abc}) = 1] \\ -&Pr[\mathcal{B}(g, g^a, g^b, g^c, \widetilde{g}, \widetilde{g}^a, \widetilde{g}^b, \widetilde{g}^c, e(g, \widetilde{g})^r) = 1]| \geq |(1/2 \pm \epsilon) - 1/2| = \epsilon \end{aligned}$$

as required.

In the following, we summarize the above statements into a bound on the advantage of the adversary in the $Game_{Receiver}$:

$$Adv_{\mathcal{A}}^{Game_{Receiver}}(\lambda) \leq Adv_{\mathbb{G}_1, \mathcal{B}}^{BDDH}(\lambda) + q_k/p + Adv^{OTS}.$$

This completes the proof of $Game_{Receiver}$. □

4.4 Our SCF-PEKS Against KG Attack

Game 3(Keyword Guessing Attack): $\mathcal{A}$ is an outside attacker (neither the server nor the receiver).

Theorem 3 *The above scheme is IND-KGA secure without random oracle model assuming that the SXDH problem is intractable.*

Proof. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, in IND-KGA Game that can attack our scheme in the standard model. We build a simulator $\mathcal{B}$ that can play a DDH (SXDH problem in $\mathbb{G}_1$) game.

We first let the challenger set the groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ with an efficient bilinear map e and a generator g of $\mathbb{G}_1$. $\mathcal{B}$ inputs a DDH instance (g^a, g^b, T), and has to distinguish $T = g^{ab}$ from a random element in $\mathbb{G}_1$.

We will start by defining the first game to be the experiment which is described in the definition of IND-KGA security.

- Game 1. Game 1 is merely the same game as the IND-KGA security experiment. While describing the experiment, we will introduce several notations and conventions to describe how the simulator selects the values that appear in its simulation.

  In the beginning of the game, the simulator selects some values a and b, which are uniformly distributed over $\mathbb{Z}_p^*$. The simulation proceeds according to the value selected. In the subsequent games, the simulator will no longer

use these values a and b, and instead, the simulator will only use g^a, g^b and $T = g^{ab}$ for its simulation of the security experiment. The values of a and b are not used in any further action, as if they were "forgotten", as these are just part of the initialization.

At some point during the game the adversary chooses a challenge keyword, we will refer to this keyword as w^*.

In the Guess Phase. The adversary continues to make its oracle queries, and the subsequent trapdoor requests must be different from the target keyword w^*. Finally, adversary $\mathcal{A}$ returns a guessing bit $\gamma' \in \{0, 1\}$. If $\gamma' = \gamma$ the simulator returns $\beta' = 1$, else it returns $\beta' = 0$. This completes the description of the simulator. Note that the simulator behaves exactly as in the original IND-KGA security experiment.

During its execution $\mathcal{A}$ may query the trapdoor oracle for some keyword w. Let q be the total number of trapdoor queries that the adversary makes. We collect all those keywords used to make queries to the trapdoor oracle in the set $\overline{W}$. Let W be the subset of queried keywords obtained by removing from $\overline{W}$ all multiples and the target keyword. We write $W = \{w^{(1)}, \cdots, w^{(q_0)}\}$ for some $q_0 \leq q$ such that $w^{(i)} \neq w^{(j)}$ for each $1 \leq i \neq j \leq q_0$ and w^* is not in W. Furthermore, we define $W^* = W \cup \{w^*\}$.

The proof of the theorem is obtained by considering subsequent games, Game 1, Game 2, Game 3 and Game 4. These games will be quite similar to Game 1. In every game. the simulator's output bit β' will be well-defined. Let X_1 be the event that $\mathcal{A}$ is successful in Game 1. Our goal is to put an upper bound on $Adv_{\mathcal{A}}^{IND-KGA}(\lambda) = |Pr[X_1] - 1/2| = |Pr[\beta' = 1] - 1/2|$. For each of the following experiments we will call X_i the event that the adversary is successful in Game i.

- Game 2. Game 2 is essentially the same as Game 1 except for the following changes:

  · To generate $(h_0, h_1, \cdots, h_n)$, the simulator the computes $m = 2q$, and randomly chooses

$$k \in \{0, \cdots, n\} \tag{4}$$
$$x', x_1, \cdots, x_n \in \{0, \cdots, p-1\} \tag{5}$$
$$y', y_1, \cdots, y_n \in \{0, \cdots, m-1\} \tag{6}$$

  Sets $h_0 = g^{x'}(g^b)^{p-km+y'}$, for $i \in \{1, \cdots, n\}$, $h_i = g^{x_i}(g^b)^{y_i}$.

  · For any keyword $w = (w_1, \cdots, w_n) \in \{0, 1\}^n$, let

$$x(w) = x' + \sum_{i=1}^{n}(x_i \cdot w_i) \tag{7}$$
$$y(w) = p - km + y' + \sum_{i=1}^{n}(y_i \cdot w_i) \tag{8}$$
$$h(w) = x(w) + by(w) \tag{9}$$

  Note that this change does not affect the distribution of the hash keys

$(h_0, h_1, \cdots, h_n)$.

Let $view_{\mathcal{A}}$ be the adversary's random tape, and the transcript of its interactions with its oracles in the current run of the experiment of Game 2. Concretely, fix all the random variables that the adversary $\mathcal{A}$ gets to see during its execution, including its random coin tosses: fix $\mathcal{G}P$, $pk_{\mathcal{R}}$ and $pk_{\mathcal{S}}$, the challenge bit γ, and the randomness used in answering the trapdoor queries. Now, the adversary can be seen as a deterministic algorithm, in particular the set of all queried (distinct) keywords $W^* = \{w^{(1)}, \cdots, w^{(q_0)}, w^*\}$ can be seen as fixed.

Let $\overline{Y} = (y', y_1, \cdots, y_n, k)$ where the random variables are distributed as described above. Clearly, if we fix $view_{\mathcal{A}}$ and re-run the experiment, the random variable $\overline{Y}$ has the same distribution as for a run of the experiment without a fixed view of the adversary. This is true due to the random "masks" x_i.

· (FA: Forced abort) We call FA the event that one of the following conditions is true. If FA occurs then the experiment is aborted and the outcome is chosen randomly.

(1) The adversary asks a trapdoor query for a keyword w such that $y(w) = 0 \bmod p$.

(2) The adversary chooses a challenge keyword w^* such that $y(w^*) \neq 0 \bmod p$.

We call this as 'forced abort' since in subsequent games, the simulator is modified such that it always has to abort once this event happens. For fixed $view_{\mathcal{A}}$, we define $\eta(view_{\mathcal{A}}) = Pr_{\overline{Y}}[\neg FA]$.

Let λ_{low} and λ_{up} be the lower bound and upper bound on $\eta(view_{\mathcal{A}})$. The following lemma bounds $\eta(view_{\mathcal{A}})$.

Lemma 3. For every fixed $view_{\mathcal{A}}$, we have

$$\lambda_{low} \equiv \frac{1}{4(n+1)q} \leq \eta(view_{\mathcal{A}}) \leq \frac{1}{2q} \equiv \lambda_{up}$$

This lemma is the same as in [22] (cf. Lemma 6.2 in [22]) and also as an extension of a lemma by Waters [33] that only proved the lower bound on $view_{\mathcal{A}}$. For completeness, we also provide the proof in Appendix A.

Compared to Game 1 we will make two step modifications to the experiment in Game 2. Since adversary $\mathcal{A}$ has already terminated, we can assume $view_{\mathcal{A}}$ to be fixed from now on.

· Step 1.(FA: Forced abort) After adversary $\mathcal{A}$ outputs his guess bit γ', the experiment checks if the event FA occurs. If yes, the experiment returns a random bit as its output bit β' and aborts. Otherwise, it continues as before. To get rid of unwanted dependence the experiment adds some artificial abort such that in total it always aborts with probability around $1 - \lambda_{low}$, independent of the view of the adversary $view_{\mathcal{A}}$.

· Step 2.(AA: Artificial Abort) After the adversary $\mathcal{A}$ outputs his guess bit γ', the experiment checks if the event FA occurs. If yes, the experiment

returns a random bit as its output bit β' and aborts. Otherwise, it continues as follows: first it samples an estimate $\eta'(view_{\mathcal{A}})$ of the probability $\eta(view_{\mathcal{A}})$ that the event FA does not happen. We want to stress that $view_{\mathcal{A}}$ is fixed at this point so sampling does not involve running adversary $\mathcal{A}$ again. By definition, this estimate $\eta'(view_{\mathcal{A}})$ is a random variable that only depends on the queried keywords W^* (and the randomness used to sample).

Depending on the estimate $\eta'(view_{\mathcal{A}})$ the simulator distinguishes two cases:

Case $\eta'(view_{\mathcal{A}}) \leq \lambda_{low}$: the simulator continues as before.

Case $\eta'(view_{\mathcal{A}}) > \lambda_{low}$: With probability $1 - \frac{\lambda_{low}}{\eta'(view_{\mathcal{A}})}$ the simulator aborts and outputs a random bit β'. With probability $\frac{\lambda_{low}}{\eta'(view_{\mathcal{A}})}$ the simulator does not abort and continues as before.

This concludes the description of Game 2. The following claim that is related to the events X_1 and X_2 will be proved in Appendix B.

Claim 1. Set $\rho(\lambda) \equiv Adv^{SXDH}_{\mathbb{G}_1,\mathcal{B}}(\lambda) \cdot q(n+1) > 0$. If the experiment takes

$$s(\lambda) = \mathcal{O}(n^2(\rho(\lambda))^{-2} ln((nq\rho(\lambda))^{-1}))$$

samples when computing the estimate $\eta'(view_{\mathcal{A}})$, then

$$|Pr[X_1] - (\frac{1}{2} + (Pr[X_2] - \frac{1}{2}) \cdot 4q(n+1))| \leq \rho(\lambda)$$

This claim is similar to [22,21,23] (cf. Lemma 6.3 in [21]). We provide the proof in Appendix B.

- Game 3. Game 3 is the same as Game 2 except that we change the way that the public keys are generated, and trapdoors are computed. Let $T = g^{ab}$ in Game 3.

(1) Setup: Let λ be the security parameter and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, \widetilde{g})$ be the bilinear map parameters. Specify a cryptographically secure pseudorandom number generator (PRNG) $H' : \mathbb{G}_T \to \mathbb{Z}_p^*$. Generators $\widetilde{u} = \widetilde{g}^{\alpha_1}$ and $\widetilde{v} = \widetilde{g}^{\alpha_2}$ for random $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$, and generators a strongly unforgeable one-time signature scheme $sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The global parameters are $\mathcal{G}P = (p, g, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \widetilde{u}, \widetilde{v}, sig, H')$.

The simulator choose $c \in \mathbb{Z}_p^*$ uniformly at random and computes $\widetilde{Q} = \widetilde{g}^c$. Let $X = g^a$, the server's public key is $pk_{\mathcal{S}} = (\widetilde{Q}, X)$ and private key $sk_{\mathcal{S}} = (a, c)$ (Note that the simulator $\mathcal{B}$ cannot know the value a but can know the value c).

The simulator $\mathcal{B}$ choose $y, z \in \mathbb{Z}_p^*$ uniformly at random and computes $\widetilde{Y} = \widetilde{g}^y, \widetilde{Z} = \widetilde{g}^z$.

The simulator $\mathcal{B}$ outputs $pk_{\mathcal{R}} = (\widetilde{Y}, \widetilde{Z}, h_0, h_1, \cdots, h_n)$ and $sk_{\mathcal{R}} = (y, z)$ as the receiver's public and private key respectively, and sends $(pk_{\mathcal{R}}, sk_{\mathcal{R}}, pk_{\mathcal{S}})$ to $\mathcal{A}$.

(2) Query phase 1. $\mathcal{A}$ makes the following trapdoor queries:

· Trapdoor query $\langle w \rangle$: If $\mathcal{A}$ queries w (such that $y(w) = 0 \bmod p$) to

the trapdoor generation oracle, then $\mathcal{B}$ chooses $s_w \in \mathbb{Z}_p^*$ and computes $d_w = X^{x(w)(z-s_w)/(y-w)}$. Let the trapdoor be $T_w = (d_w, s_w)$. Sends T_w to $\mathcal{A}$.

(3) Challenge. Once $\mathcal{A}$ decides that phase 1 is over, it outputs a keyword pair (w_0, w_1). $\mathcal{B}$ responds by choosing a random $\gamma \in \{0, 1\}$, and let the challenge keyword $w^* = w_\gamma$. Choose $s_{w^*} \in \mathbb{Z}_p^*$ and compute

$$d_{w^*} = \left(X^{x(w^*)} T^{y(w^*)}\right)^{(z-s_w^*)/(y-w^*)}$$

Let the trapdoor $T_{w^*} = (d_{w^*}, s_{w^*})$. Sends the challenge trapdoor T_{w^*} to $\mathcal{A}$.

As in Game 2, $y(w^*) \neq 0 \bmod p$, then $H(w^*) = g^{x(w^*)+by(w^*)}$. We have

$$d_{w^*} = \left(X^{x(w^*)} T^{y(w^*)}\right)^{(z-s_w^*)/(y-w^*)}$$

$$= \left(X^{x(w^*)} X^{by(w^*)}\right)^{(z-s_w^*)/(y-w^*)} = X^{h(w^*)(z-s_w^*)/(y-w^*)}$$

(4) Query phase 2. $\mathcal{A}$ issues a number of trapdoor queries as in phase 1. The restriction here is that w_0 and w_1 are not allowed to be queried as trapdoor queries.

(5) Guess. $\mathcal{A}$ outputs the guess γ'.

Following a technical argument it is easy to check that the public key and trapdoors in games 2 and 3 are distributed identically. Thus, the probabilities of success in both games are equal:

$$Pr[X_2] = Pr[X_3]$$

- Game 4. Game 4 is the same as Game 3 except that the value $T = g^{ab}$ is replaced by a random value $T \in \mathbb{G}_1$ which is chosen at the beginning. Let $T = g^{ab}$ in Game 3.

$$Claim\ 2.\ |Pr[X_3] - Pr[X_4]| \leq Adv^{SXDH}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$$

Proof. The idea of the proof is the following. We are given g, g^a, g^b and T which is either a random element of $\mathbb{G}_1$ or g^{ab}. We simulate the adversary. When $T = g^{ab}$ then we simulate the adversary perfectly in Game 3. When T is uniform in $\mathbb{G}_1$ then we simulate the adversary in Game 4. Thus, if the adversary distinguishes between games 3 and 4, then we distinguish between the two possible values for T with the same probability.

Finally, since T is uniform in $\mathbb{G}_1$ then we have

$$Pr[X_4] = 1/2.$$

Analysis. We now summarize the above statements into a bound on the advantage of the adversary in the IND-KGA game:

$$
\begin{aligned}
Adv_{\mathcal{A}}^{IND-KGA}(\lambda) &= |Pr[X_1] - 1/2| \\
&\leq |(Pr[X_2] - 1/2) \cdot 4q(n+1)| + \rho(\lambda) \\
&= |(Pr[X_3] - 1/2)| \cdot 4q(n+1) + \rho(\lambda) \\
&\leq (|Pr[X_4] - 1/2| + Adv_{\mathbb{G}_1,\mathcal{B}}^{SXDH}(\lambda)) \cdot 4q(n+1) + \rho(\lambda) \\
&= Adv_{\mathbb{G}_1,\mathcal{B}}^{SXDH}(\lambda) \cdot 4q(n+1) + \rho(\lambda)
\end{aligned}
$$

Note that $\rho(\lambda) \equiv Adv_{\mathbb{G}_1,\mathcal{B}}^{SXDH}(\lambda) \cdot q(n+1)$ then we obtain

$$
Adv_{\mathcal{A}}^{IND-KGA}(\lambda) \leq Adv_{\mathbb{G}_1,\mathcal{B}}^{SXDH}(\lambda) \cdot 5q(n+1)
$$

□

5 Performance Comparison

In this section, we compare our scheme with the existing secure channel-free public key encryption with keyword search schemes in [10,27,30]. Let $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|\mathbb{G}_T|$, $|svk|$ and $|\sigma|$ denote the bit-length of an element in groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, the verification key and signature of one-time signature, respectively. We denote t_p, t_e, t_s, and t_v as the computational cost of a bilinear pairing operation, an exponentiation over a bilinear group, a one-time signature and verification, respectively. Let svk and σ be the one-time signatures public key and signature. The result of the comparison is outlined in Table 2.

Scheme		**Rhee et al. [30]**	**Rhee et al.**[27]	**Fang et al. [10]**	**Ours**
Cost	PEKS	$t_p + 2t_e$	$9t_p + 2t_e$	$3t_p + 4.5t_e$	$3t_p + 6.5t_e + t_s$
	Test	$t_p + 2t_e$	$t_p + t_e$	$4t_p + 3t_e + t_v$	$4t_p + 3t_e + t_v$
Length	$PEKS$	$\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$\|\mathbb{G}_1\| + \|\mathbb{G}_T\|$	$3\|\mathbb{G}_1\| + 2\|\mathbb{G}_T\|$	$\|svk\| + \|\mathbb{G}_1\| + 2\|\mathbb{G}_2\| + 2\|\mathbb{G}_T\| + \|\sigma\|$
	Trapdoor	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_p^*\|$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_p^*\|$
IND-CKA		Yes	Yes	Yes	Yes
IND-KGA		Yes	No	No	Yes
Without ROM		No	No	Yes	Yes
Test query		No	Weaker	No	Yes

Table 2. Performance Comparison Among Various SCF-PEKS Schemes

6 Applications of SCF-PEKS

In this section, for completeness, we provide some applications for SCF-PEKS schemes. We specifically select an application to demonstrate how SCF-PEKS schemes can be used to present solutions in these scenarios.

Public-key Encryption with Keyword Search (PEKS) is a mechanism that allows search by keywords on encrypted data. It aims at preserving the privacy of the receiver of the data while providing a way that allows her to search efficiently without the need of decrypting the ciphertext.

Thus, public key encryption with keyword Search (PEKS) has become more important, as it has recently been applied to various applications, such as storage and retrieval of encrypted sensitive data in cloud computing, privacy-preserving online photo sharing, privacy-preserving location based sharing(LBS) [9], Encrypted and Searchable Audit Log [34], Personal Health Record (PHR) and intelligent email routing [1,5,2].

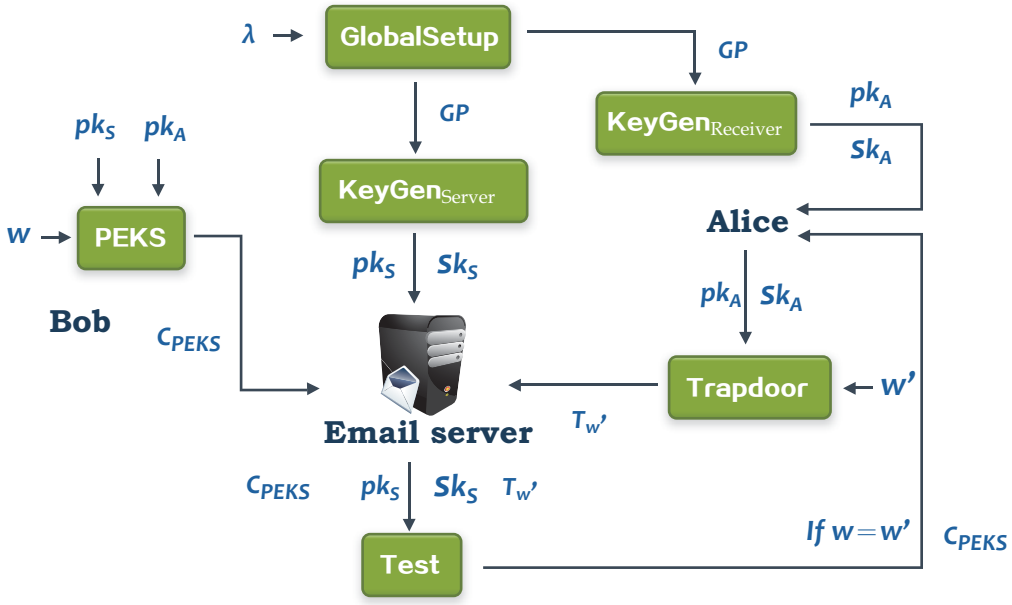

Figure 1. Application in intelligent email routing

In Figure 1, Bob sends a ciphertext $\widetilde{C}$, $\widetilde{C} = (C_{PKE} \parallel C_{PEKS}) = (PKE(pk_A, m) \parallel PEKS(pk_A, w))$, to Alice where pk_A is Alice's public key, C_{PKE} is an encrypted version of Bob's message under pk_A and w is the keyword that Bob wants to attach to the email (for example "urgent"). Alice can provide the mail server with a certain trapdoor T_w (which is a trapdoor constructed by Alice on a keyword w) through a public channel that enables the mail server to test whether the encrypted keyword associated with the message (C_{PEKS}) is equal to the word w of Alice selected. Given $PEKS(pk_A, w')$ and T_w, the server can test whether $w \stackrel{?}{=} w'$. If $w = w'$, then the mail server learns nothing more about w'.

7 Conclusion

In this paper, we provide a formal model of SCF-PEKS secure against keyword guessing attacks. Furthermore, we present an SCF-PEKS scheme secure against chosen keyword and ciphertext attacks, and keyword guessing attacks. Based on the DBDH assumption, SXDH assumption and the truncated q-ABDHE assumption, we first proved its indistinguishability of secure channel free PEKS against chosen keyword and ciphertext attack (IND-SCF-CKCA) security without random oracle. We also analyzed the computational consistency and security against keyword guessing attacks (IND-KGA) of our scheme.

This work motivates a few interesting questions. First, how to achieve a more efficient SCF-PEKS scheme without random oracle is desirable. We opened the new direction on how to achieve this notion, but the more efficient variant is certainly required. Second, how to construct SCF-PEKS scheme secure against keyword guessing attacks without requiring bilinear pairing operations would be very interesting.

Acknowledgement

The second author is supported by ARC Future Fellowship FT0991397. This paper was supported in part by the National Natural Science Foundation of China (NSFC)(Grant No. 61272083 and 61139002). We also thank Congjun Wang for his help to improve the clarity of this paper.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE and Extensions. In *Proc. of The 25th Annual International Cryptology Conference, Advances in Cryptology-CRYPTO 2005*, LNCS 3621, Springer-Verlag, 2005, pp. 205–222.

[2] J. Baek, R. Safavi-Naini and W. Susilo. Public Key Encryption with Keyword Search Revisited. In *Proc. of Applied Cryptography and Information Security 06, ACIS 2006*, LNCS 5072, Springer-Verlag, 2008, pp. 1249 - 1259.

[3] J. Baek, R. Safavi-Naini and W. Susilo. On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search. In *Proc. of 9th Information Security Conference, ISC 2006*, LNCS 4176, Springer-Verlag, 2006, pp. 217 - 232.

[4] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G Persiano. Public Key Encryption with Keyword Search. In *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology-EUROCRYPT 2004*. LNCS 3027, Springer-Verlag, 2004, pp. 506–522.

[5] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Proc. of Fourth IACR Theory of Cryptography Conference, TCC 2007*. LNCS 4392, Springer-Verlag, 2007, pp. 535–554.

[6] J.W. Byun, H.S. Rhee, H. A. Park, and D.H. Lee. Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. In *Proc. of 3rd VLDB Workshop on Secure Data Management, SDM 2006,* . LNCS 4165, Springer-Verlag, 2006, pp. 75–83.

[7] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. In *Proc. of the Thirtieth Annual ACM Symposium on the Theory of Computing, STOC 1998*, ACM Press, 1998, pp. 209–218.

[8] R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. In *EUROCRYPT 2004*, LNCS 3027, Springer, Heidelberg, 2004, pp. 202–222.

[9] C. Dong and N. Dulay. Longitude: A Privacy-preserving Location Sharing Protocol for Mobile Applications. In *Proc. of IFIPTM 2011*. IFIP AICT 358, pp. 133-148 .(2011)

[10] L. Fang, W. Susilo, C. Ge and J. Wang. A Secure Channel Free Public Key Encryption With Keyword Search Scheme Without Random Oracle. In *Proc. of Cryptology and Network Security, 8th International Conference, CANS 2009*, LNCS 5888, Springer, Heidelberg, 2009, pp. 248–258.

[11] R.R. Farashahi, B. Schoenmakers, and A. Sidorenko. Efficient pseudorandom generators based on the DDH assumption. In *Proc. of PKC 2007*, LNCS 4450, Springer, Heidelberg, 2007, pp. 426–441.

[12] T. Fuhr and P. Paillier. Decryptable searchable encryption. In *Proc. of First International Conference on Provable Security, ProvSec 2007*, LNCS 4784, Springer-Verlag, 2007, pp. 228–236.

[13] C. Gentry. Practical identity-based encryption without random oracles. In *Proc. of 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology-EUROCRYPT 2006*, LNCS 4004, Springer-Verlag, 2006, pp. 457–464.

[14] C. Gu,Y Zhu, and H. Pan. Efficient Public Key Encryption with Keyword Search Schemes from Pairings. In *Proc. of Information Security and Cryptology: Third SKLOIS Conference, Inscrypt 2007*, LNCS 4990, Springer-Verlag, 2007, pp. 372–383.

[15] P. Golle, J. Staddon, and B. Waters. Secure Conjunctive Search over Encrypted Data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.), In *Proc. of Second International Conference on Applied Cryptography and Network Security, ACNS 2004*. LNCS 3089,Springer-Verlag, 2004, pp. 31–45.

[16] M. Green and S. Hohenberger. Universally Composable Adaptive Oblivious Transfer. In *Proc. of of The 14th Annual International Conference on the Theory and Application of Cryptology and Information Security, ASIACRYPT 2008*. LNCS 5350, Springer-Verlag, 2008, pp. 179–197.

[17] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proc. of 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology-EUROCRYPT 2008*. LNCS 4965, Springer-Verlag, 2008, pp. 415–432.

[18] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*. Express, Vol. 58, No. 301, 1963, pp. 13–30.

[19] D. Hofheinz and E. Weinreb. Searchable encryption with decryption in the standard model. *Cryptology ePrint Archive, Report 2008/423*. 2008. http://eprint.iacr.org/2008/423.

[20] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee. Constructing PEKS schemes secure against keyword guessing attacks is possible? *Computer Communications*. Express, Vol. 32, No. 2, 2009, pp.394–396.

[21] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. In *Proc. of Information Security and Privacy, 11th Australasian Conference, ACISP 2006*, LNCS 4058, Springer-Verlag, 2006, pp. 336–347.

[22] E. Kiltz and D. Galindo. Direct chosen-ciphertext secure identity-based key encapsulation without random oracles. *Theoretical Computer Science*, Elsevier. Vol. 410, No. 47-49, 2009, pp. 5093–5111.

[23] E. Kiltz and Y. Vahlis. CCA2 Secure IBE: Standard Model Efficiency through Authenticated Symmetric Encryption. In *Proc. of RSA Conference 2008, Cryptographers' Track, CT-RSA 2008*, LNCS 4964, Springer-Verlag, 2008, pp. 221–238.

[24] T.Okamoto, K.Sakurai, H.Shizuya, How intractable is the discrete logarithm for a general finite group? In *Proc. Eurocrypt 1992*, Lecture Notes in Computer Science 658, Springer-Verlag, Berlin (1992) 420–428.

[25] D.J. Park, K. Kim, P.J. Lee. Public Key Encryption with Conjunctive Field Keyword Search. In: Lim, C.H., Yung, M. (eds.), In *Proc. of Information Security Applications, 5th International Workshop, WISA 2004*. LNCS 3325, Springer-Verlag, 2005, pp. 73–86.

[26] C. Petit, F. Standaert, O. Pereira, T. G. Malkin, M. Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In *Proc. of the 3th international Symposium on information, Computer, and Communications Security, ASIACCS 2008*, ACM, New York, NY, 2008, pp. 56–65.

[27] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Improved searchable public key encryption with designated tester. In *Proc. of the 4th international Symposium on information, Computer, and Communications Security, ASIACCS 2009*, ACM, New York, NY, 2009, pp. 376–379.

[28] H. S. Rhee, J. H. Park, and D. H. Lee. Generic construction of designated tester public-key encryption with keyword search, Information Sciences. Express, Vol. 205, No. 1, 2012, pp. 93–109.

[29] H. S. Rhee, W. Susilo and H-J. Kim. Secure searchable public key encryption scheme against keyword guessing attacks. *IEICE Electron.* Express, Vol. 6, No. 5, 2009, pp.237–243.

[30] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, Express, Vol. 83, No. 5, 2010, pp. 763–771.

[31] J. Shao, Z. Cao, X. Liang, H. Lin, Proxy re-encryption with keyword search, Information Sciences. Express, Vol. 180, No. 13, 2010, pp. 2576–2587.

[32] U.V. Vazirani and V.V. Vazirani. Efficient and secure pseudo-random number generation. In *Proc. of 25th Annual Symposium on Foundations of Computer Science*, LNCS 3494, Springer-Verlag, 1984, pp. 458–63.

[33] B. Waters. Efficient identity-based encryption without random oracles. In *Proc. of 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Advances in Cryptology-EUROCRYPT 2005*, LNCS 3494, Springer-Verlag, 2005, pp. 114–127.

[34] B. Waters, D. Balfanz, G. Durfee, D. Smetters. Building an Encrypted and Searchable Audit Log. *Proc. of Network and Distributed System Security Symposium, NDSS 2004*, (2004)

[35] W. C. Yau, S. H. Heng and B. Goi. Off-Line Keyword Guessing Attacks on Recent Public Key Encryption with Keyword Search Schemes. In *Proc. of The 5th International Conference on Autonomic and Trusted Computing, ATC 2008*, LNCS 5060, Springer-Verlag, 2008, pp. 100–105.

[36] B. Zhang, F. Zhang. An efficient public key encryption with conjunctive-subset keywords search, *Journal of Network and Computer Applications.* Express, Vol. 34, No. 1, pp. 262-267. (2011)

[37] R. Zhang and H. Imai. Generic combination of public key encryption with keyword search and public key encryption. In *Proc. of Cryptology and Network Security, 6th International Conference, CANS 2007*, LNCS 4856, Springer-Verlag, 2007, pp. 159–174.

Appendix A: Proof of Lemma 3.

Fix $view_{\mathcal{A}}$ and hence the queried keywords $W^* = \{w^{(1)}, \cdots, w^{(q_0)}, w^*\}$. We abbreviate $\eta = \eta(view_{\mathcal{A}})$. For an integer t, define the event

$$E(t) : \bigwedge_{i=1}^{q_0} (y(w^{(i)}) \neq 0 \ mod \ t) \wedge y(w^*) = 0 \ mod \ t.$$

With this notation recall that $\eta = Pr_{\overline{Y}}[E(p)]$ and we intend to show that

$$\lambda_{low} \equiv \frac{1}{4(n+1)q} \leq \eta \leq \frac{1}{2q} \equiv \lambda_{up}$$

(Also recall that $\overline{Y} = (y', y_1, \cdots, y_n, k)$, where the random variables $(y', y_1, \cdots, y_n, k)$ are distributed as in Eq. (6).). Over the integers we have $y(w^*) = p - km + y' + \sum_{i=1}(y_i \cdot w_i^*)$ where $0 \leq y' + \sum_{i=1}(y_i \cdot w_i^*) < (n+1)m < p$. Sets $k^* = \lfloor \frac{y' + \sum_{i=1}(y_i \cdot w_i^*)}{m} \rfloor$, then $y(w^*) = p + (k^* - k)m$ where $0 \leq k, k^* < n+1$, *i.e.,* $0 \leq (k^* - k)m < (n+1)m < p$. This shows that if $y(w^*) = 0 \ mod \ m$, then there is a unique $0 \leq k < n+1$ such that $y(w^*) = 0 \ mod \ p$. If $k^* = k$ and $y(w^*) = 0 \ mod \ m$, then clearly $y(w^*) = 0 \ mod \ p$. On the other hand, if $y(w^*) \neq 0 \ mod \ m$ then $y(w^*) \neq 0 \ mod \ p$. Hence,

$$\begin{aligned}
\eta = Pr_{\overline{Y}}[E(p)] &\geq Pr[k = k^*] Pr_{\overline{Y}}[E(p) \mid k = k^*] \\
&= \frac{1}{n+1} Pr_{\overline{Y}}[E(p) \mid k = k^*] \\
&\geq \frac{1}{n+1} Pr_{\overline{Y}}[E(m) \mid k = k^*] \\
&= \frac{1}{n+1} Pr_{\overline{Y'}}[E(m) \mid k = k^*]
\end{aligned}$$

where the probability space $\overline{Y'}$ contains the random variables $(y', y_1, \cdots, y_n)$ distributed according to Eq. (6), for fixed k, *i.e.,* from now on we consider k to be fixed. Define $Pr_{\overline{Y'}}[E(m)] = \eta_m$. Since trivially $\eta_m \geq \eta$, we obtain

$$\frac{1}{n+1}\eta_m \leq \eta \leq \eta_m$$

Next, we will compute an upper and lower bound on η_m. Let $w \neq w'$ and $a, b \in \mathbb{Z}$. We collect some simple observations on function $y(\cdot)$ which essentially show that the $y(\cdot)$ mod m are pairwise independent:

$$Pr_{\overline{Y'}}[y(w) = b \ mod \ m] = \frac{1}{m} \tag{10}$$

$$Pr_{\overline{Y'}}[y(w) = a \ mod \ m \mid y(w') = b \ mod \ m] = \frac{1}{m} \tag{11}$$

Eq. (10) follows since for any choice of $(y', y_1, \cdots, y_n, k)$ there is a single choice of y' that will make the condition hold. To show Eq. (11) assume there exists an index $1 \leq i \leq n$ such that $w_i = 1$ and $w'_i = 0$. Then fix all y_j's for $j \neq i$ except y_i so that $y(w') = b$. Therefore $Pr_{\overline{Y'}}[y(w) = a \mid y(w') = b] = \frac{1}{m}$. If there is no such i then we can use Bayes to reverse roles of w and w'.

We continue to bound η_m with

$$\begin{aligned}
\eta_m &= Pr_{\overline{Y'}}[\bigwedge_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \ mod \ m) \mid y(w^*) = 0 \ mod \ m] Pr_{\overline{Y'}}[y(w^*) = 0 \ mod \ m] \\
&= \frac{1}{m} Pr_{\overline{Y'}}[\bigwedge_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \ mod \ m) \mid y(w^*) = 0 \ mod \ m] \\
&= \frac{1}{m}(1 - Pr_{\overline{Y'}}[\bigvee_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \ mod \ m) \mid y(w^*) = 0 \ mod \ m]) \\
&\geq \frac{1}{m}(1 - \sum_{i=1}^{q_0} Pr_{\overline{Y'}}[(y(w^{(i)}) \neq 0 \ mod \ m) \mid y(w^*) = 0 \ mod \ m]) \\
&= \frac{1}{m}(1 - \sum_{i=1}^{q_0} \frac{1}{m}) \\
&\geq \frac{1}{m}(1 - \frac{q}{m}) \\
&= \frac{1}{4q}
\end{aligned}$$

$$\begin{aligned}
\eta_m &= \frac{1}{m}(1 - Pr_{\overline{Y'}}[\bigvee_{i=1}^{q_0}(y(w^{(i)}) \neq 0 \ mod \ m) \mid y(w^*) = 0 \ mod \ m]) \\
&\leq \frac{1}{m} = \frac{1}{2q}
\end{aligned}$$

where the last equation follows by our choice of $m = 2q$ which minimizes the term. Thus, we obtain

$$\frac{1}{4(n+1)q} \leq \frac{1}{n+1}\eta_m \leq \eta \leq \eta_m \leq \frac{1}{2q}$$

□

Appendix B: Proof of Claim 1.

Let AA be the event that the experiment artificially aborts at the end of the simulation. Let total abort $TA = FA \bigvee AA$ be the event that it aborts artificially or forced. We abbreviate $\rho = \rho(\lambda)$ and $s = s(\lambda)$. First we claim

Claim 2. For any fixed $view_{\mathcal{A}}$, $|Pr[\neg TA] - \lambda_{low}| \leq \frac{\lambda_{low}\rho}{2}$

The proof of the claim is postponed until later. Since the claim holds for any fixed $view_{\mathcal{A}}$ it also remains true for random $view_{\mathcal{A}}$, conditioned on $\gamma' = \gamma$ and $\gamma' \neq \gamma$:

$$|Pr[\neg TA \mid \gamma' = \gamma] - \lambda_{low}| \leq \frac{\lambda_{low}\rho}{2} \tag{12}$$

$$|Pr[\neg TA \mid \gamma' \neq \gamma] - \lambda_{low}| \leq \frac{\lambda_{low}\rho}{2} \tag{13}$$

We continue computing $Pr[X_2]$:

$$\begin{aligned} Pr[X_2] &= Pr[\beta' = 1 \wedge TA] + Pr[\beta' = 1 \wedge \neg TA] \\ &= Pr[\beta' = 1 \mid TA](1 - Pr[\neg TA]) + Pr[\beta' = 1 \wedge \neg TA] \end{aligned}$$

In case of abort the simulator outputs a random bit β'. If the simulator does not abort then it outputs $\beta' = 1$ if $\gamma' = \gamma$. Therefore we can continue with

$$\begin{aligned} Pr[X_2] &= Pr[\beta' = 1 \mid TA](1 - Pr[\neg TA]) + Pr[\beta' = 1 \wedge \neg TA] \\ &= \frac{1}{2}(1 - Pr[\neg TA]) + Pr[\gamma' = \gamma \wedge \neg TA] \\ &= \frac{1}{2}(1 - (Pr[\gamma' \neq \gamma \wedge \neg TA] + Pr[\gamma' = \gamma \wedge \neg TA])) + Pr[\gamma' = \gamma \wedge \neg TA] \\ &= \frac{1}{2} + \frac{1}{2}(Pr[\gamma' = \gamma \wedge \neg TA] - Pr[\gamma' \neq \gamma \wedge \neg TA]) \\ &= \frac{1}{2} + \frac{1}{2}(Pr[\neg TA \mid \gamma' = \gamma]Pr[\gamma' = \gamma] - Pr[\neg TA \mid \gamma' \neq \gamma]Pr[\gamma' \neq \gamma]) \end{aligned}$$

Since $Pr[\gamma' = \gamma] = Pr[X_1]$ we obtain

$$\begin{aligned} Pr[X_2] - \frac{1}{2} &= \frac{1}{2}(Pr[\neg TA \mid \gamma' = \gamma]Pr[X_1] - Pr[\neg TA \mid \gamma' \neq \gamma](1 - Pr[X_1])) \\ &= \frac{1}{2}Pr[X_1](Pr[\neg TA \mid \gamma' = \gamma] + Pr[\neg TA \mid \gamma' \neq \gamma]) - \frac{1}{2}Pr[\neg TA \mid \gamma' \neq \gamma]. \end{aligned}$$

Combining this with Eqn. (12) and (13) , we obtain

$Pr[X_2] - \frac{1}{2} - \lambda_{low}(Pr[X_1] - \frac{1}{2}) = \frac{1}{2}Pr[X_1]((Pr[\neg TA \mid \gamma' = \gamma] - \lambda_{low}) + (Pr[\neg TA \mid \gamma' \neq \gamma] - \lambda_{low})) - \frac{1}{2}(Pr[\neg TA \mid \gamma' \neq \gamma] - \lambda_{low})$.

$$\begin{aligned}|Pr[X_2] - \frac{1}{2} - \lambda_{low}(Pr[X_1] - \frac{1}{2})| &\leq \frac{1}{2}Pr[X_1](\frac{\lambda_{low}\rho}{2} + \frac{\lambda_{low}\rho}{2}) + \frac{\lambda_{low}\rho}{2} \\ &\leq \lambda_{low}\rho\end{aligned}$$

where the last inequality stems from $0 \leq Pr[X_1] \leq 1$. To prove the Claim 1 it leaves to prove Claim 2.

Lemma 4. (Hoeffding's Bound [18]). Let $X_1, X_2, \cdots X_s$ be independent random variables with $a \leq X_i \leq b$ and define $\overline{X} = \frac{1}{s}\sum_{i=1} X_i$. Then, for any $t > 0$, we have the inequality

$$Pr[|\overline{X} - E(\overline{X})| \geq t] \leq 2e^{-2s(\frac{t}{b-a})^2}$$

where $E(\overline{X})$ denotes the expected values of $\overline{X}$.

Proof of Claim 2. We abbreviate $\rho = \rho(\lambda)$, $\eta = \eta(view_{\mathcal{A}})$ and $\eta' = \eta'(view_{\mathcal{A}})$. By construction the two events AA and FA are independent and consequently we have

$$Pr[\neg TA] = Pr[\neg FA]Pr[\neg AA] = \eta Pr[\neg AA]$$

Set $0 < \rho' = \frac{1}{8}\rho \leq \frac{1}{8}$. We make

$$s(\lambda) = 8q^{-2}(\rho\lambda_{low})^{-2}ln((\frac{1}{16}\rho\lambda_{low})^{-1}) = \mathcal{O}(n^2(\rho(\lambda))^{-2}ln((nq\rho(\lambda))^{-1}))$$

samples to compute an approximation η' of η. For each sample we pick independently according to the distribution $\overline{Y}$ from Eq. (6), each indicator variable X_i is defined as: If FA, $X_i = 0$ and otherwise $X_i = 1$.

By construction, $\lambda_{low} \leq Pr[X_i = 1] \leq \lambda_{up}$. Finally, we make a majority decision over all X_i by computing $\eta' = \frac{1}{s(\lambda)}\sum_{i=1}^{s(\lambda)}(X_i)$. By construction, $E(\eta') = \eta \geq \lambda_{low}$. Using Hoeffding's Bound (cf. Lemma 6.8, [22]) for the estimation η' of η, with $t = \eta\rho'$ and $b - a \leq \lambda_{up} - \lambda_{low} \leq \frac{1}{2q}$

$$
\begin{aligned}
Pr[|\eta' - \eta| \geq \eta\rho'] &\leq 2e^{-2s(\frac{\eta\rho\cdot 2\cdot q}{8})^2} \\
&\leq 2e^{-2s(\frac{\lambda_{low}\rho q}{4})^2} \\
&\leq 2e^{-2\cdot 8q^{-2}(\rho\lambda_{low})^{-2}ln((\frac{1}{16}\rho\lambda_{low})^{-1})(\frac{\lambda_{low}\rho q}{4})^2} \\
&\leq \frac{\lambda_{low}\rho}{8}.
\end{aligned}
$$

Since $\rho' = \frac{\rho}{8}$. We call the approximation η' as "GOOD" if $|\eta' - \eta| \leq \frac{\eta\rho}{8}$ and "BAD" otherwise. Then the above establishes

$$Pr[\eta' BAD] \leq \lambda_{low}\rho'.$$

For every fixed good η' we have $\eta(1-\rho') \leq max\{\lambda_{low}, \eta'\} \leq \eta(1+\rho')$. Since $Pr[\neg AA] = \frac{\lambda_{low}}{max\{\lambda_{low},\eta'\}}$, we have

$$\frac{\lambda_{low}}{\eta(1+\rho')} \leq Pr[\neg AA \mid \eta' GOOD] \leq \frac{\lambda_{low}}{\eta(1-\rho')}.$$

We provide an upper bound on $Pr[\neg TA]$.

$$
\begin{aligned}
Pr[\neg TA] &= \eta(Pr[\neg AA \mid \eta' GOOD]Pr[\eta' GOOD] + Pr[\neg AA \mid \eta' BAD]Pr[\eta' BAD]) \\
&\leq \eta(\frac{\lambda_{low}}{\eta(1-\rho')} + \lambda_{low}\rho') \\
&\leq \lambda_{low}(\frac{1}{\eta(1-\rho')} + \rho') \\
&\leq \lambda_{low}(1+4\rho') = \lambda_{low}(1+\frac{1}{2}\rho).
\end{aligned}
$$

This completes the proof. $\square$