SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

March 2020, Vol. 63 132102:1–132102:16
https://doi.org/10.1007/s11432-019-1509-7

An enhanced searchable encryption scheme for secure data outsourcing

Rui ZHANG[1,2], Jiabei WANG[1,2*], Zishuai SONG[1,2] & Xi WANG[1,2]

[1]*State Key Laboratory of Information Security (SKLOIS), Institute of Information Engineering, Chinese Academy of Sciences, Beijing* 100093*, China;*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing* 100049*, China*

Received 15 April 2019/Accepted 2 August 2019/Published online 10 February 2020

Abstract In the cloud environment, where the cloud server cannot always be fully trusted, both data and query privacy should be well protected for secure data outsourcing. Searchable encryption provides a more practical solution to secure data storage while enabling efficient search queries. In this paper, four important problems of public key encryption with keyword search (PEKS), namely, a scheme without secure channels, conjunctive keyword search, (offline) outside and inside keyword guessing attack (full KGA) resistance and proof in the standard model, are considered. We provide an in-depth analysis of the reasons behind (offline) full KGA by considering two types of PEKS schemes as examples. In particular, we introduce the concept of server-aided secure channel free public key encryption with conjunctive keyword search (SA-SCF-PECKS) which can resist (offline) full KGA. Furthermore, we provide a concrete and efficient construction of SA-SCF-PECKS, and prove its security in the standard model. To the best of our knowledge, our proposal is the first PECKS scheme to address these four problems simultaneously. We compare the security and efficiency of our scheme with those of other related PECKS schemes in theoretical and practical ways. In general, compared with other schemes, our SA-SCF-PECKS scheme shows better performance in terms of security and efficiency.

Keywords secure data outsourcing, (offline) outside and inside keyword guessing attack, secure channel free PEKS, conjunctive keyword search, server-aided scheme, standard model

Citation Zhang R, Wang J B, Song Z S, et al. An enhanced searchable encryption scheme for secure data outsourcing. Sci China Inf Sci, 2020, 63(3): 132102, https://doi.org/10.1007/s11432-019-1509-7

1 Introduction

1.1 Motivation

Compared to centralized data storage, which may cause larger accidents, cloud storage provides a relatively inexpensive and user-friendly solution to manage big data for enterprises, governments, and organizations, while also accelerates the wide proliferation of sensitive data.

The recent revelation of the largest data leak in Swedish history in 2017 reignited government and user concerns on data privacy, particularly when dealing with data in the cloud. This leakage which was mainly caused by the improper outsourcing deal between the Swedish transport agency (STA) and IBM indicates that data privacy protection is necessary in the cloud computing environment and cloud servers that can be controlled by adversaries cannot always be fully trusted. The cloud server may spy on users' data and attempt to learn more about user queries. Moreover, lots of examples also indicate that systematic approaches, such as firewalls and access control, do not inherently protect data privacy, particularly

* Corresponding author (email: wangjiabei@iie.ac.cn)

against adversaries within the system. Traditional encryption protects data privacy at the expense of data utility, such as keyword search, which is a basic database functionality. Thus, a solution that securely outsources data and allows efficient queries meanwhile is clearly needed. Queries must be well protected because they may reveal insights of querier's interests, agendas, modes of operation. Such protection is particularly crucial when handling government affairs or data with high security requirements.

Herein, we design a practical solution to secure data outsourcing with enhanced privacy protection. Both user data and query privacy are well protected in our solution, which also supports conjunctive keyword search on encrypted data. Ideally, fully homomorphic encryption [1] or ORAM [2] can help us design such protocols that leak absolutely no information to the cloud server. Unfortunately, these approaches are impractical that even slower than downloading the encrypted data and doing search locally by user himself [3]. The proposal of searchable encryption introduces a more practical solution for the above problem. The notion of public key encryption with keyword search (PEKS) was first proposed by Boneh et al. [4] and designed for the store-and-forward system; it was later extended to many other scenarios. The data owner (e.g., government, enterprise, or person) may want to outsource confidential documents to the public cloud, and the querier who may be the data owner himself or some other authorized users, can query the cloud server for documents containing a specific keyword. Traditional PEKS, is exempt from sophisticated secret key management and has flexible application scenarios. Indeed, the following issues must be addressed as practical or theoretical requirements in secure data outsourcing.

Firstly, the secure channel between the cloud server and querier required in traditional PEKS scheme is expensive and overburdened for the case with weak infrastructure. As mentioned in [5], adopting some mature techniques, such are SSL/TLS, to build a secure channel still incurs heavy computation and communication costs, that renders this type of schemes weakly adaptable.

Secondly, supporting single keyword search on encrypted data is far from meeting application requirements. Thus, we desire to design a PEKS scheme supporting conjunctive keyword search (PECKS) [6].

Thirdly, the commonly-queried keyword space usually has low entropy because the owner's data is very likely related to some specific areas. Thus, (outside or inside) adversaries may be able to guess the queried keywords and crack these keywords within a reasonable short period of time. This limitation is an inherent security flaw in traditional PEKS called (offline) keyword guessing attack (KGA). More specifically, an attacker with the querier's public key can guess a possible keyword and encrypt it to generate the corresponding ciphertext. The attacker can then test whether this ciphertext matches the target trapdoor. The above process can be repeated until the correct underlying keyword is found, thereby breaking the trapdoor privacy in PEKS. If the attacker is from outside of the cloud services system, we call it (offline) outside keyword guessing attack (OKGA). If the attacker is the cloud server, system administrator or other insiders within the cloud provider, we call it (offline) inside keyword guessing attack (IKGA). We say it is secure against full KGA, when a scheme can go against both OKGA and IKGA. In Sweden's data breach, the unmodified dataset was outsourced to IBM, who, in turn, outsourced the data storage and management to subcontractors with full access permissions. In this case, the use and management of data were completely out of the data owner's (STA's) control. A malicious cloud server can do anything it wishes. Thus, privacy risks remain when a PEKS scheme applied to this scenario cannot go against IKGA.

Finally, constructing a scheme in the standard model is usually more desirable, and the same for constructing a more secure PEKS scheme. The scheme with random oracles may never be secure when it is instantiated in the standard model [7].

Motivated by the above scenarios, we aim to construct a PEKS scheme for secure data outsourcing that takes all of these four issues into consideration which are more secure and practical. Besides, this construction can be easily integrated into cloud platform, such as ownCloud.

1.2 Related work

Several directions have been further studied following the first PEKS proposed by Boneh et al. [4].

Secure channel free. Baek et al. [5] considered a new variant of PEKS called secure channel free

Table 1 Security comparison of various PECKS schemes

	GSW04 [9]	PKL05 [6]	CH09 [11]	ZZ11 [13]	HHL14 [24]	YM16 [27]	MML17 [14]	ZW17 [26]	Ours
Trapdoor unforgeability	✓	✓	✓	✓	✓	✓	✓	✓	✓
Ciphertext anonymity	✓	✓	✓	✓	✓	✓	✓	✓	✓
Secure channel free	×	×	×	×	✓	✓	✓	✓	✓
Outside KGA	✓	×	×	×	×	✓	✓	✓	✓
Inside KGA	×	×	×	×	×	×	×	×	✓
Standard model	×	×	×	✓	✓	✓	✓	✓	✓

PEKS (SCF-PEKS). The cloud server in SCF-PEKS possesses its own public/private key pair, so that trapdoors can be transferred without a secure channel. In later work, Rhee et al. [8] strengthened the security model of Baek et al.'s SCF-PEKS.

Conjunctive keyword search. Overall, existing PECKS schemes can be categorized into two types: the fixed keyword field scheme and the variable keyword field scheme. In the fixed keyword field scheme [6, 9–12], m fixed keyword fields must be defined for each document. Two assumptions are still necessary [9]. Under these assumptions, the null field must be padded with some symbols, which has poor impact on efficiency. Besides, the querier must determine the keywords to search, as well as the corresponding keyword fields. This condition is impractical to extend to other scenarios involving various documents. While the variable keyword field scheme [13, 14] requires a smaller storage space of the cloud server, and keywords can be listed in any order, which is more flexible and efficient.

KGA resistance. KGA was first observed by Byun et al. [15]. Fang et al. [16], Xu et al. [17] and Guo et al. [18] later attempted to construct PEKS schemes that are secure against OKGA, but these schemes still did nothing to inside attackers. Wang et al. [19] creatively set two cloud servers without collusion in their framework to resist IKGA. Chen et al. [20] applied a similar idea and proposed a new model called dual-server public key encryption with keyword search (DS-PEKS) that could resist IKGA. However, because the keyword search process was performed separately by two servers, DS-PEKS scheme may still suffer from the search inefficiency. Huang et al.'s [21] public-key authentication encryption with keyword search (PAEKS) solved IKGA by embedding the data owner's private key into the searchable ciphertexts. Jiang et al.'s work (SEK-IA) [22] integrated identity-based encryption (IBE) into PEKS to resist inside attackers. Both PAEKS and SEK-IA applied the similar idea that added data owner's authentication information into the searchable ciphertexts, such that the trapdoor produced in their schemes should also contain the data owner's specified identity (the public key or the identity in IBE). Therefore, when PAEKS is applied to the multi-owner setting, the data user must create a specific trapdoor for each data owner, and the number of trapdoors is linear to the number of data owners. Although the SEK-IA scheme has a constant-sized trapdoor even in the multi-owner scenario, the cloud server must search for each data owner's ciphertexts, and the trapdoor changes as the specified data owner identity set changes. An additional trusted third party is needed in the SEK-IA scheme. Besides, the above two solutions require major modifications to the original PEKS scheme, and the scenario scalability is affected. Sun et al. [23] combined a signcryption algorithm and indistinguishability obfuscator (io) to deal with IKGA in PEKS, which, unfortunately, inherits the impracticability of io. PECKS schemes also suffer from this type of attack. Hwang et al. [24] showed two previous PECKS schemes that are insecure against OKGA and constructed a SCF-PECKS scheme against OKGA. However, Lu et al. [25] showed that Hwang et al.'s scheme [24] is still susceptible to full KGA by giving three concrete KGAs. Zhao et al.'s scheme [26] and Miao et al.'s VCKSM scheme [14] also fail to resist IKGA. Therefore, devising a secure SCF-PECKS that can resist full KGA remains an unsolved problem.

Proof in the standard model. A comparison of some related PECKS schemes is shown in Table 1 [6, 9, 11, 13, 14, 24, 26, 27]. To the best of our knowledge, no known scheme that meets all the requirements in Subsection 1.1 simultaneously is yet available. In this paper, we address all of the problems described above, and propose an enhanced searchable encryption scheme that can resist full KGA, the security of this scheme is proven in the standard model.

1.3 Our contributions

Firstly, we point out a recent SCF-PEKS scheme and a recent PECKS scheme that have failed to resist (offline) IKGA. We note that the same holds for most existing PEKS schemes (including the PEKS scheme and its variants). Secondly, we propose a new framework with enhanced privacy protection, called server-aided secure channel free public key encryption with conjunctive keyword search (SA-SCF-PECKS), which can stand against full KGA. Thirdly, we construct a concrete SA-SCF-PECKS scheme in the standard model. To the best of our knowledge, it is the first PECKS scheme that can resist full KGA and can be proven secure in the standard model. This scheme can be constructed in an environment without a secure channel between the cloud server and the querier which is highly practical. Finally, we compare the efficiency of our scheme with those of other related PECKS schemes in theory and practice. We implemented our SA-SCF-PECKS scheme and the public key encryption with conjunctive-subset keywords search (PECSK) scheme in ZZ11 [13]. Moreover, our scheme can be integrated into cloud storage applications, such as ownCloud.

2 Preliminary

In this section, we review some useful notations and notions.

Notations. $r \xleftarrow{\$} S$ denotes uniformly choosing a random element r from a set S. $S_1 \to S_2$ means a mapping from set S_1 to set S_2. We also define $\{s_i\}_{i=1}^{k} := \{s_1, \ldots, s_i, \ldots, s_k\}$. $|S|_b$ denotes the bit-length of an element in a set S. A function $\mathrm{negl}(n)$: $\mathbb{N} \to \mathbb{R}$ is negligible in n if for every positive polynomial $p(\cdot)$ there exists an N such that $\forall\ n > N$, $\mathrm{negl}(n) < 1/p(n)$.

Bilinear pairing. Let G_1, G_2, and G_T be three multiplicative groups of prime order p, and g_1 be a generator of G_1, g_2 be a generator of G_2. We say $e : G_1 \times G_2 \to G_T$ is a bilinear map, if the following three conditions are met:

- Bilinearity: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$, $\forall a, b \in \mathbb{Z}_p$.
- Non-degeneracy: $e(g_1, g_2) \neq 1_{G_T}$.
- Computability: $\forall g_1 \in G_1$ and $\forall g_2 \in G_2$, $e(g_1, g_2)$ can be computed efficiently.

We use Type-III bilinear groups in our concrete construction.

The decisional Diffie-Hellman (DDH) assumption. Consider a cyclic group G of prime order p, and with a generator g. The DDH problem is to distinguish the tuples of (g^a, g^b, g^{ab}) and (g^a, g^b, g^c) where $a, b, c \xleftarrow{\$} \mathbb{Z}_p$. We define the advantage $\mathrm{Adv}_{G,\mathcal{A}}^{\mathrm{DDH}}(\cdot)$ of an adversary $\mathcal{A}$ as $|\Pr[\mathcal{A}(g^a, g^b, g^{ab}) = 1] - \Pr[\mathcal{A}(g^a, g^b, g^c) = 1]|$. The DDH assumption holds if $\mathrm{Adv}_{G,\mathcal{A}}^{\mathrm{DDH}}(\cdot)$ is negligible for all PPT $\mathcal{A}$.

Blind signature. We briefly review the definition of blind signature. In our scheme we use a deterministic round-optimal blind signature (deBS), we will adapt the definition to this case.

(1) Syntax. Our deterministic blind signature deBS scheme includes five polynomial-time algorithms, executed by a signer and a user.

- $\mathrm{KeyGen}_{\mathrm{deBS}}(1^\lambda)$: Taking as input a security parameter λ, the algorithm returns a parameter P_{deBS} of deBS, a public/private key pair $(\mathrm{pk}_{\mathrm{deBS}}, \mathrm{sk}_{\mathrm{deBS}})$ of the signer and a secret key k_U for the user. Suppose λ and P_{deBS} are the default inputs to the following algorithms in deBS scheme.
- User-Request$(\mathrm{pk}_{\mathrm{deBS}}, k_U, m)$: Taking as input a public key $\mathrm{pk}_{\mathrm{deBS}}$, a secret key k_U, and a message m, this algorithm generates a user's signature request ξ, with state st.
- Signer-Issue$(\mathrm{sk}_{\mathrm{deBS}}, \xi)$: Taking as input the signer's private key $\mathrm{sk}_{\mathrm{deBS}}$ and a user's signature request ξ, it outputs a pre-signature $\bar{\sigma}$.
- User-Process$(\mathrm{pk}_{\mathrm{deBS}}, \bar{\sigma}, \mathrm{st})$: Taking as input a public key $\mathrm{pk}_{\mathrm{deBS}}$, a pre-signature $\bar{\sigma}$, and the state st, it generates a blind signature σ_m of message m, or outputs $\perp$ if $\bar{\sigma}$ is not valid.
- $\mathrm{Verify}_{\mathrm{deBS}}(\mathrm{pk}_{\mathrm{deBS}}, m, \sigma_m)$: Taking as input a public key $\mathrm{pk}_{\mathrm{deBS}}$ of singer, a message m, and a signature σ_m, if σ_m is a valid signature of m, it returns 1; else returns 0.

Throughout our paper, blind signature scheme is deterministic, that is for each $\mathrm{pk}_{\mathrm{deBS}}$, each m, and a given user's secret key k_U, only one valid signature σ_m exists s.t.$\mathrm{Verify}_{\mathrm{deBS}}(\mathrm{pk}_{\mathrm{deBS}}, m, \sigma_m) = 1$.

(2) Security requirements. One-more unforgeability and blindness are usually required.

(One-more) unforgeablity states that for all probabilistic polynomial time (PPT) adversaries, the probability of forging q_k+1 valid signatures of q_k+1 different messages after at most q_k interactions with an honest signer is negligible. Here, we need an extended concept over message set. Blindness means the probability of judging the order in which two messages m_0 and m_1 are signed, for an adversarial signer, after interacting with an honest user, is negligible.

3 Inside keyword guessing attacks (IKGAs) on some previous schemes

We now present IKGA on an SCF-PEKS scheme [28] and a PECKS scheme [13], respectively. A brief review of these schemes can be found in Appendix. We note that this attack is not considered in their proposals and this rather shows that many previous schemes do not fulfill this security requirement.

3.1 IKGA against an SCF-PEKS scheme

Fang et al.'s scheme is constructed in the standard model, the details of which are described in Appendix B. We note that the potential IKGA vulnerability is outside of the original security model of [28].

Lemma 1. Fang et al.'s scheme [28] is vulnerable to (offline) IKGA.

Proof. If there exists a potential insider attacker $\mathcal{A}$, who is the server or other insider within the cloud management system without server's private key sk_S, $\mathcal{A}$ can launch (offline) IKGA as detailed below:

Step 1. Attacker $\mathcal{A}$ selects his target trapdoor $T_w = (r_{T_w}, d_{T_w})$ from the trapdoors received from querier, where $d_{T_w} = (r_{q,2} \cdot g^{-r_{T_w}})^{\frac{1}{r_{q,1}-w}}$, $r_{T_w} \xleftarrow{\$} \mathbb{Z}_p^*$.

Step 2. $\mathcal{A}$ guesses a possible keyword w^* according to his knowledge background.

Step 3.

- If $\mathcal{A}$ is the server, it can just use the public key of querier and server, to generate the ciphertext C_w^* of the keyword w^*. Then it takes its own private key sk_s, the ciphertext C^* and the target trapdoor T_w to test if C^* and T_w match. If the test passed, $\mathcal{A}$'s guessed keyword w^* is the underlying keyword of T_w. Otherwise, go back to Step 2.

- Even if $\mathcal{A}$ does not have the server's private key sk_S, it can use the public key $\mathrm{pk}_{\mathrm{QU}}$ of the querier and the guessed keyword w^*, to check if $e(d_{T_w}, q \cdot g^{-w^*}) = e(g, r_{q,2} \cdot g^{-r_{T_w}})$ holds. If so, the guessed keyword w^* is the underlying keyword of T_w. Else, go back to Step 2.

Commonly-used keyword space in each specific field (such as government, finance) has low entropy. Thus, the inside attacker can easily launch the above attack within a short period of time.

3.2 IKGA against a PECKS scheme

In this subsection, we consider a PECSK scheme [13] (as shown in Appendix B) which is also a variable keyword field one.

Lemma 2. Zhang et al.'s scheme [13] is vulnerable to (offline) IKGA.

Proof. Assuming $\mathcal{A}$ is a potential insider attacker, the (offline) IKGA can be conducted as below:

Step 1. First, $\mathcal{A}$ selects a target trapdoor $T_Q = (T_0, T_1, \ldots, T_m, (g_{G_1,1})^{r_t}, q^{r_t})$.

Step 2. $\mathcal{A}$ guesses a possible keyword set $W^* = \{w_1^*, w_2^*, \ldots, w_t^*\}$.

Step 3. $\mathcal{A}$ can carry out this attack in two ways.

- Taking $\mathrm{pk}_{\mathrm{QU}}$ of the querier and keyword set W^*, $\mathcal{A}$ can generate the ciphertext C_{W^*} of keyword set W^*, then do the Test himself to check if it outputs "1". If so, such keyword set W^* is the correct guess corresponding to T_Q. Otherwise, go back to Step 2.

- More easily, with the querier's public key $\mathrm{pk}_{\mathrm{QU}}$ and the guessed keyword set W^*, $\mathcal{A}$ can check if $e(T_j, q) = e(g_{G_1,2}, g_{G_2}) \cdot e((g_{G_1,1})^{r_t}, p_j \cdot g_{G_2}) \cdot e((g_{G_1,1})^{(H_1(w_1^*)^j + H_1(w_2^*)^j + \cdots + H_1(w_t^*)^j)/t}, p_1 \cdot g_{G_2})$, where $0 \leqslant j \leqslant m$. If so, the guessed keyword set W^* is the underlying keyword set of the trapdoor T_Q. Otherwise, go back to Step 2.

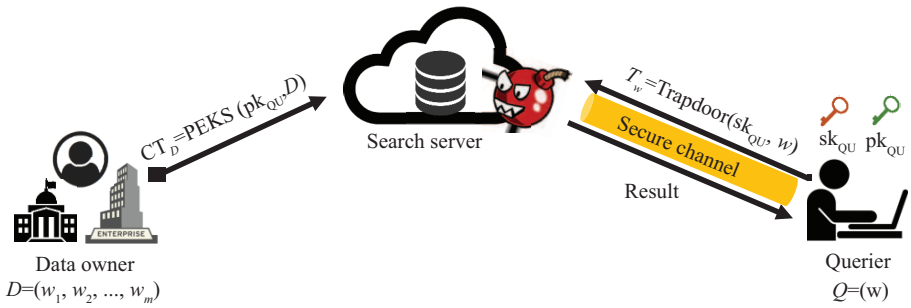

Figure 1 (Color online) Traditional PEKS system model.

Because the queried keyword space often has low entropy, $\mathcal{A}$ can perform the above KGA successfully.

Other PECKS schemes such as Hwang et al.'s scheme [24] claim to be able to resist offline OKGA, but cannot resist IKGA. Once a malicious server receives the trapdoor from the querier, it can generate a PECKS ciphertext of the guessed keyword set using its own public key and the public key of the querier and then verify it. Designing a PECKS scheme that is secure against full KGA is of great significance. Three main reasons can be summarized for the offline IKGA.

- For the inside attacker, the trapdoor can be obtained easily.
- In real applications, commonly-used keywords are always selected from a low-entropy keyword space, it is feasible for the attacker with some background knowledge to launch keyword guessing attack.
- The ciphertext generation process is out of the querier and data owner's control, i.e., the ciphertext is derived directly from the original keyword (or the keyword set). Thus, an attacker with the public keys of the target roles can compute the ciphertext of its guess. Moreover, the inside attacker can perform the Test freely. Both algorithms (Ciphertext generation and Test) can be performed offline.

4 Model of server-aided secure channel free PECKS

In this section, we present the system model and the security definition for SA-SCF-PECKS.

4.1 System model

From the observations in Section 3, we transform the offline ciphertext generation process into an online one. Motivated by the basic idea for the single keyword search scheme against KGA set forth by Chen et al. [29], we add a semi-honest keyword server (KS) to assist with keyword preprocessing, which is essential, before generating the PECKS ciphertext and trapdoor. A system model comparison of the traditional PEKS and our scheme is shown in Figures 1 and 2. Our model contains three types of roles: the adversarial user (data owner or querier) who is usually weak in storage and computing, and would like to outsource his or her data to a cloud server and obtain the corresponding services from it, the adversarial SS(search server) which provides data storage services for data owners and responds to querier's search requests, and the semi-honest KS, who engages in the interactive protocol with the user to generate valid preprocessed keyword set (PKW). An unmentioned trusted key generation center (KGC) which is responsible for initializing system parameters and generating keys for corresponding roles, is also needed.

We consider that no secure channel is required between the SS and the querier in our system model. In our model, the user is adversarial and may try to forge a valid PKW based on the PKWs that he previously obtained from the KS. Our scheme is secure for the honest party even in this scenario.

4.2 Formal definition

In the SA-SCF-PECKS scheme, we focus on how to generate index according to the keyword fields of each document. We assume there are m different keywords in each document and denote the keyword set of the document as $D = (w_1, w_2, \ldots, w_m)$. We also write the queried keyword set as $Q = (w'_1, w'_2, \ldots, w'_t)$,

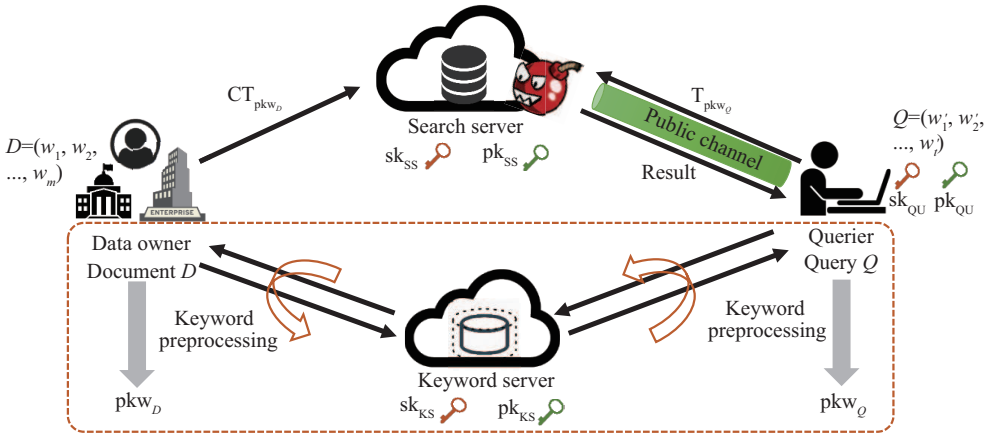

Figure 2 (Color online) Our SA-SCF-PECKS system model.

where t is the number of keywords in the query Q. Our scheme is a variable keyword field scheme, as shown in Figure 2.

Definition 1 (SA-SCF-PECKS). A SA-SCF-PECKS scheme $\Pi_{\text{SA-SCF-PECKS}}$ is a tuple of the following polynomial-time algorithms such that:

- GlobalSetup(1^λ): Taking as input a security parameter λ, it outputs a global parameter $\mathcal{GP} = (P_{\text{deBS}}, P_{\text{PECKS}})$, where P_{deBS} and P_{PECKS} are parameters of deBS and PECKS, respectively.
- SA-KeyGen$_U$(1^λ): Taking as input a security parameter λ, it outputs a secret key k_U of a user (data owner and querier share the same secret key).
- SA-KeyGen$_{\text{KS}}$(P_{deBS}): Taking as input a parameter P_{deBS} of deBS, it outputs a public/private key pair of KS as $(\text{pk}_{\text{KS}}, \text{sk}_{\text{KS}})$.
- SA-KeyGen$_{\text{SS}}$(P_{PECKS}): Taking as input a parameter P_{PECKS} of PECKS, it outputs a public/private key pair of SS as $(\text{pk}_{\text{SS}}, \text{sk}_{\text{SS}})$.
- SA-KeyGen$_{\text{QU}}$(P_{PECKS}): Taking as input a parameter P_{PECKS} of PECKS, it outputs a public/private key pair of querier as $(\text{pk}_{\text{QU}}, \text{sk}_{\text{QU}})$.
- SA-PKW($P_{\text{deBS}}, k_U, \text{pk}_{\text{KS}}, \text{sk}_{\text{KS}}, W$): Taking as input a parameter P_{deBS} of deBS, a secret key k_U of a User, a key pair $(\text{pk}_{\text{KS}}, \text{sk}_{\text{KS}})$ of KS and a keyword set W (a document D or a conjunctive keyword query Q), it returns the preprocessed keyword set of W as pkw_W.
- SA-dPECKS($P_{\text{PECKS}}, \text{pk}_{\text{SS}}, \text{pk}_{\text{QU}}, \text{pkw}_D$): Taking as input a parameter P_{PECKS} of PECKS, a public key pk_{SS} of SS, a public key pk_{QU} of querier, and a preprocessed keyword set pkw_D, the algorithm returns the PECKS ciphertext of D as CT_{pkw_D}.
- SA-dTrapdoor($P_{\text{PECKS}}, \text{pk}_{\text{SS}}, \text{sk}_{\text{QU}}, \text{pkw}_Q$): Taking as input a parameter P_{PECKS} of PECKS, a public key pk_{SS} of SS, a private key sk_{QU} of querier, and a preprocessed keyword set pkw_Q, it generates the trapdoor of Q as T_{pkw_Q}.
- SA-dTest($P_{\text{PECKS}}, \text{sk}_{\text{SS}}, T_{\text{pkw}_Q}, \text{CT}_{\text{pkw}_D}$): Taking as input a parameter P_{PECKS} of PECKS, a private key sk_{SS} of SS, a trapdoor T_{pkw_Q}, and a PECKS ciphertext CT_{pkw_D}, it outputs "True" if $Q \subseteq D$ or "False" otherwise.

The first five algorithms are the initial system setup and key generation, which are executed by the trusted KGC. The algorithm SA-PKW is a protocol between the User and the KS. The algorithms SA-dPECKS, SA-dTrapdoor, and SA-dTest are executed by the data owner, the querier and the SS, respectively.

4.3 Security model

We set new security models for the outside attacker (OA), the adversarial search server (KS), the semi-honest keyword server (KS), and the adversarial user (data owner or querier), respectively.

Outside attacker (OA). The notion of indistinguishability against keyword guessing attack (IND-KGA) introduces the trapdoor indistinguishability, it guarantees that an OA who has obtained the trapdoors of the challenged keyword sets will still be unable to deduce the underlying PKW (or original keyword set) of the trapdoor, i.e., OA cannot perform the OKGA.

Definition 2 ($\mathrm{Exp}^{\text{IND-KGA}}_{\mathcal{A}(\text{OA})}$)**.** The IND-KGA experiment $\mathrm{Exp}^{\text{IND-KGA}}_{\mathcal{A}(\text{OA})}(1^\lambda)$ for any $\Pi_{\text{SA-SCF-PECKS}}$, PPT outside adversary $\mathcal{A}$ and security parameter λ is defined as follows:

• Setup. The challenger calls the algorithm $\mathrm{GlobalSetup}(1^\lambda)$ to generate the global parameters $\mathcal{GP}$, then runs the algorithm SA-KeyGen_U, $\text{SA-KeyGen}_{\text{KS}}$, $\text{SA-KeyGen}_{\text{QU}}$ and $\text{SA-KeyGen}_{\text{SS}}$ to generate the corresponding keys and sends $(\mathrm{pk}_{\text{QU}}, \mathrm{pk}_{\text{KS}}, \mathrm{pk}_{\text{SS}})$ to $\mathcal{A}$.

• Query-I (Trapdoor query). $\mathcal{A}$ could ask the challenger the corresponding trapdoor of any keyword set Q. The challenger responds to it by performing $\mathrm{pkw}_Q \leftarrow \text{SA-PKW}(P_{\text{deBS}}, k_U, \mathrm{pk}_{\text{KS}}, \mathrm{sk}_{\text{KS}}, Q)$ and $T_{\mathrm{pkw}_Q} \leftarrow \text{SA-dTrapdoor}(P_{\text{PECKS}}, \mathrm{pk}_{\text{SS}}, \mathrm{sk}_{\text{QU}}, \mathrm{pkw}_Q)$.

• Challenge. $\mathcal{A}$ chooses two keyword sets Q_0, Q_1 which have not be queried by $\mathcal{A}$ in Query-I stage and gives them to the challenger. Then, the challenger chooses $b \in \{0,1\}$ randomly, generates $\mathrm{pkw}_{Q_b} \leftarrow \text{SA-PKW}(P_{\text{deBS}}, k_U, \mathrm{pk}_{\text{KS}}, \mathrm{sk}_{\text{KS}}, Q_b)$, $T^* \leftarrow \text{SA-dTrapdoor}(P_{\text{PECKS}}, \mathrm{pk}_{\text{SS}}, \mathrm{sk}_{\text{QU}}, \mathrm{pkw}_{Q_b})$ and sends T^* to $\mathcal{A}$.

• Query-II (Trapdoor query). $\mathcal{A}$ again issues the trapdoors of the keyword sets, but the queries on the challenge keyword sets Q_0, Q_1 are not allowed.

• Output. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. The experiment returns 1 if $b' = b$ which means $\mathcal{A}$ wins, and 0 otherwise.

$\mathcal{A}$'s winning advantage is denoted as $\mathrm{Adv}^{\text{IND-KGA}}_{\mathcal{A}(\text{OA})}(1^\lambda) = |\Pr[\mathrm{Exp}^{\text{IND-KGA}}_{\mathcal{A}(\text{OA})}(1^\lambda) = 1] - 1/2|$.

Adversarial search server (SS). We give the definition of indistinguishability against chosen keyword guessing attack (IND-CKGA) here to guarantee that both PECKS ciphertext and trapdoor do not leak the corresponding keywords to SS. Different from the case with an OA, the adversarial SS is allowed to obtain the ciphertext and trapdoor pair of the same challenge keyword set. IND-CKGA also guarantees that the SS cannot perform the offline IKGA.

Definition 3 ($\mathrm{Exp}^{\text{IND-CKGA}}_{\mathcal{A}(\text{SS})}$)**.** The IND-CKGA experiment $\mathrm{Exp}^{\text{IND-CKGA}}_{\mathcal{A}(\text{SS})}(1^\lambda)$ for any $\Pi_{\text{SA-SCF-PECKS}}$, PPT inside adversary (the SS) $\mathcal{A}$ and security parameter λ is defined as follows:

• Setup. The challenger calls the algorithm $\mathrm{GlobalSetup}(1^\lambda)$ to generate the global parameters $\mathcal{GP}$, then runs the algorithm SA-KeyGen_U, $\text{SA-KeyGen}_{\text{KS}}$, $\text{SA-KeyGen}_{\text{QU}}$ and $\text{SA-KeyGen}_{\text{SS}}$ to generate the corresponding keys and sends $(\mathrm{pk}_{\text{QU}}, \mathrm{pk}_{\text{KS}}, \mathrm{sk}_{\text{SS}}, \mathrm{pk}_{\text{SS}})$ to $\mathcal{A}$.

• Query-I (Ciphertext query & Trapdoor query). $\mathcal{A}$ could ask the challenger the PECKS ciphertext and trapdoor of any keyword set $W_{q_{\text{I}}}$. Assume $\mathcal{A}$ has queried q_{I_k} distinct keyword sets $\{W_1, W_2, \ldots, W_{q_{\text{I}_k}}\}$ in the Query-I stage.

• Challenge. $\mathcal{A}$ chooses the challenge keyword sets W_0^*, W_1^* which then will be sent to the challenger. We limit that $W_b^* \cap (W_1 \cup \cdots \cup W_{q_{\text{I}_k}}) = \phi$, where $b \in \{0,1\}$. For W_0^*, W_1^*, the challenger selects $b \in \{0,1\}$ randomly and generates $\mathrm{pkw}_{W_b^*} \leftarrow \text{SA-PKW}(P_{\text{deBS}}, k_U, \mathrm{pk}_{\text{KS}}, \mathrm{sk}_{\text{KS}}, W_b^*)$, $\mathrm{CT}^* \leftarrow \text{SA-dPECKS}(P_{\text{PECKS}}, \mathrm{pk}_{\text{SS}}, \mathrm{pk}_{\text{QU}}, \mathrm{pkw}_{W_b^*})$, $T^* \leftarrow \text{SA-dTrapdoor}(P_{\text{PECKS}}, \mathrm{pk}_{\text{SS}}, \mathrm{sk}_{\text{QU}}, \mathrm{pkw}_{W_b^*})$ and sends (CT^*, T^*) to $\mathcal{A}$.

• Query-II (Ciphertext query & Trapdoor query). $\mathcal{A}$ again asks for the ciphertext and trapdoor of any keyword set $W_{q_{\text{II}}}$. Assume $\mathcal{A}$ has queried q_{II_k} distinct keyword sets $\{W_1, W_2, \ldots, W_{q_{\text{II}_k}}\}$ in the Query-II stage, we require that any keyword set $W_{q_{\text{II}}} \cap (W_1^* \cup W_0^*) = \phi$.

• Output. $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. The experiment returns 1 if $b' = b$ which means $\mathcal{A}$ wins, and 0 otherwise.

$\mathcal{A}$'s winning advantage is denoted as $\mathrm{Adv}^{\text{IND-CKGA}}_{\mathcal{A}(\text{SS})}(1^\lambda) = |\Pr[\mathrm{Exp}^{\text{IND-CKGA}}_{\mathcal{A}(\text{SS})}(1^\lambda) = 1] - 1/2|$.

Semi-honest keyword server (KS). We introduce the definition of indistinguishability against chosen keyword attack (IND-CKA) for the consideration that the interactive keyword preprocessing involving by User and KS should not leak User's private input (i.e., the original keyword set) to the semi-honest KS (or outside snoopers).

Definition 4 ($\mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}(\text{KS})}$)**.** The IND-CKA experiment $\mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}(\text{KS})}(1^\lambda)$ for any $\Pi_{\text{SA-SCF-PECKS}}$, PPT adversary (the KS) $\mathcal{A}$ and security parameter λ is defined as follows:

• Setup. The challenger first initializes the system global parameters $\mathcal{GP}$ by running algorithm GlobalSetup(1^λ), then runs the algorithm SA-KeyGen$_U$, SA-KeyGen$_{\text{KS}}$ and sends $\mathcal{A}$ the key pair (pk_{KS}, sk_{KS}). $\mathcal{A}$ then sends two challenge keyword sets W_0, W_1 to the challenger.

• Challenge. Upon receiving W_0, W_1, the challenger selects a random bit $b \in \{0,1\}$, interacts with $\mathcal{A}$ in the keyword preprocessing protocol to get the PKW pkw_{W_b} of W_b, and send pkw_{W_b} to $\mathcal{A}$.

• Output. Finally, $\mathcal{A}$ outputs its guess $b' \in \{0,1\}$. The experiment returns 1 if $b' = b$ which means $\mathcal{A}$ wins, and 0 otherwise.

$\mathcal{A}$'s winning advantage is denoted as $\mathrm{Adv}^{\text{IND-CKA}}_{\mathcal{A}(\text{KS})}(1^\lambda) = |\Pr[\mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}(\text{KS})}(1^\lambda) = 1] - 1/2|$.

Adversarial users. In our system model, we require only KS is able to produce valid PKWs, or else the scheme would be insecure against IKGA. This protocol is also expected to prevent the adversarial user from forging a valid PKW based on the PKWs received from the KS before. We define a definition called one-more unforgeability under chosen keyword attack (OMU-CKA) for this consideration.

Definition 5 ($\mathrm{Exp}^{\text{OMU-CKA}}_{\mathcal{A}(U)}$)**.** The OMU-CKA experiment $\mathrm{Exp}^{\text{OMU-CKA}}_{\mathcal{A}(U)}(1^\lambda)$ for any $\Pi_{\text{SA-SCF-PECKS}}$, PPT adversary (the user) $\mathcal{A}$ and security parameter λ is defined as follows:

• Setup. The challenger first generates the global parameters $\mathcal{GP}$, the secret key k_U and the key pair $(\text{pk}_{\text{KS}}, \text{sk}_{\text{KS}})$ by running the algorithm GlobalSetup(1^λ), SA-KeyGen$_U$ and SA-KeyGen$_{\text{KS}}$, respectively. $\mathcal{A}$ is given k_U and pk_{KS}.

• PKW-Query. $\mathcal{A}$ can adaptively issue the corresponding PKWs of at most q_k different unprocessed keyword sets $W_1, W_2, \ldots, W_{q_k}$ of his choice.

• Output. Finally, $\mathcal{A}$ outputs q_k pairs $\{W_i, \text{pkw}_{W_i}\}_{i=1}^{q_k}$ and another pair $\{\overline{W}, \text{pkw}_{\overline{W}}\}$. The experiment returns 1 if (1) $W_i \neq W_j$, $\forall i, j \in [1, q_k]$ where $i \neq j$, (2) $\overline{W} \cap (W_1 \cup \cdots \cup W_i \cup \cdots \cup W_{q_k}) = \phi$, and (3) $\forall i \in [1, q_k]$, pkw_{W_i} and $\text{pkw}_{\overline{W}}$ are all valid PKWs, which means $\mathcal{A}$ wins, and 0 otherwise.

The adversary $\mathcal{A}$'s advantage of winning the above experiment is denoted as $\mathrm{Adv}^{\text{OMU-CKA}}_{\mathcal{A}(U)}(1^\lambda)$.

Finally, combining Definitions 2–5, the security model of SA-SCF-PECKS is shown in Definition 6.

Definition 6 (Secure SA-SCF-PECKS)**.** A SA-SCF-PECKS scheme $\Pi_{\text{SA-SCF-PECKS}}$ is secure if for any PPT attacker $\mathcal{A}_i$ $(i = 1, 2, 3, 4)$, the advantages $\mathrm{Adv}^{\text{IND-KGA}}_{\mathcal{A}_1(\text{OA})}(1^\lambda)$, $\mathrm{Adv}^{\text{IND-CKGA}}_{\mathcal{A}_2(\text{SS})}(1^\lambda)$, $\mathrm{Adv}^{\text{IND-CKA}}_{\mathcal{A}_3(\text{KS})}(1^\lambda)$, and $\mathrm{Adv}^{\text{OMU-CKA}}_{\mathcal{A}_4(U)}(1^\lambda)$ are all negligible in the security parameter λ.

5 SA-SCF-PECKS scheme in the standard model

5.1 Overview

The most common and convenient method to avoid expensive secure channels is to let the cloud server keep a key pair of its own [30]. We apply this basic idea in our scheme. To enable the querier to perform conjunctive keyword search on the encrypted documents efficiently and flexibly, we construct a variable keyword field scheme. We then modify the construction for shared multi-owner settings in [14] which can only resist OKGA.

In our scheme, upon giving a keyword set W, the user first performs mutual authentication with KS (via HTTP, for example) and then runs a protocol with KS to get PKW pkw_W. Later, the data owner (or querier) generates the corresponding PECKS ciphertext (or trapdoor) of the PKW. Some requirements should be met by the interactive protocol. Firstly, the protocol must take the private information of the KS so that an adversary is unable to forge a valid PKW. Secondly, the KS should know nothing about the input keyword set of the User and the PKW's validity can be checked efficiently. Finally, the protocol transforming the original keyword set to the preprocessed one should be deterministic which means if we have $W_1 = W_2$, then $\text{pkw}_{W_1} = \text{pkw}_{W_2}$. The first two requirements ensure the security of keywords while the last one is to guarantee the correctness of SA-SCF-PECKS scheme. In this way, the inside attacker in our scheme can no longer generate the PECKS ciphertext of his guess offline. Besides, the KS can also apply some mechanisms to monitor abnormal requests or control the online KGA rate.

Deterministic blind signatures with unforgeability and blindness can well capture the above requirements. We transform the round-optimal blind signature scheme in [31] into a deterministic one (as defined in Section 2) by pseudorandom functions (PRFs). The transformed scheme (deBS) is also correct, (one-more) unforgeable and perfectly blind in the standard model (more details are provided in Appendix A). The scheme can achieve desirable efficiency and has a very low communication overhead on both sides which is crucial for users to reduce their costs in the keyword preprocessing. Our solution is fairly general and requires slight modification to the original PECKS scheme only, which allows it have better scalability than PAEKS [21].

5.2 Concrete SA-SCF-PECKS scheme $\Pi_{\text{SA-SCF-PECKS}}$

We now present our concrete construction of $\Pi_{\text{SA-SCF-PECKS}}$.

• GlobalSetup(1^λ): Taking as input the security parameter λ, the algorithm generates parameters $P_{\text{PECKS}} = (p, g_1, g_2, G_1, G_2, G_T, e_1, h)$, $P_{\text{deBS}} = (\hat{p}, g, \hat{g}, G, \hat{G}, T, e_2, F^1, F^2)$, where G_1, G_2, G_T are groups with prime order p; $G, \hat{G}, T$ are groups with prime order $\hat{p}$; g_1, g_2, g, and $\hat{g}$ are the generators of group G_1, G_2, G and $\hat{G}$, respectively. $h : \{0,1\}^* \to \mathbb{Z}_P^*$ is a cryptographic collision-resistant hash function, and F^1, F^2 are two PRFs for the User and KS, respectively. We use the bilinear map $e_1 : G_1 \times G_2 \to G_T$ and $e_2 : G \times \hat{G} \to T$. Finally, it sets the global parameters $\mathcal{GP}$ as $\mathcal{GP} = (P_{\text{deBS}}, P_{\text{PECKS}})$.

• SA-KeyGen$_U(1^\lambda)$: The algorithm selects a value $k_U \xleftarrow{\$} \{0,1\}^\lambda$. The data owner and querier share the same key k_U.

• SA-KeyGen$_{\text{KS}}(P_{\text{deBS}})$: The algorithm selects values $\bar{h}, x, y \xleftarrow{\$} \mathbb{Z}_P^*$, $k_{\text{KS}} \xleftarrow{\$} \{0,1\}^\lambda$ and sets private key $\text{sk}_{\text{KS}} = (\bar{h}, x, y, k_{\text{KS}})$. Then, it computes public key $\text{pk}_{\text{KS}} = (H, \hat{H}, \hat{X}, \hat{Y}) = (g^{\bar{h}}, \hat{g}^{\bar{h}}, \hat{g}^x, \hat{g}^y)$. Finally, it outputs the KS's public/private key pair $(\text{pk}_{\text{KS}}, \text{sk}_{\text{KS}})$.

• SA-KeyGen$_{\text{SS}}(P_{\text{PECKS}})$: The algorithm selects values $a_{\text{SS}} \xleftarrow{\$} G_2$, $\alpha \xleftarrow{\$} \mathbb{Z}_P^*$ and sets the private key $\text{sk}_{\text{SS}} = \alpha$. Then it computes public key $\text{pk}_{\text{SS}} = (a_{\text{SS}}, b_{\text{SS}}) = (a_{\text{SS}}, g_1^\alpha)$. Finally, it outputs SS's public/private key pair $(\text{pk}_{\text{SS}}, \text{sk}_{\text{SS}})$.

• SA-KeyGen$_{\text{QU}}(P_{\text{PECKS}})$: The algorithm selects a value $\beta \xleftarrow{\$} \mathbb{Z}_P^*$ and sets private key $\text{sk}_{\text{QU}} = \beta$. Then it computes public key $\text{pk}_{\text{QU}} = g_1^\beta$. Finally, it outputs querier's public/private key pair $(\text{pk}_{\text{QU}}, \text{sk}_{\text{QU}})$.

• SA-PKW$(P_{\text{deBS}}, k_U, \text{pk}_{\text{KS}}, \text{sk}_{\text{KS}}, W)$: The algorithm is the keyword preprocessing protocol between the user (data owner or querier) and the aided KS. At the end of the protocol, the user acquires the valid processed keyword set pkw_W of $W = (w_1, \ldots, w_k)$. W can be a document $D = (w_1, \ldots, w_m)$ of the data owner or a conjunctive keyword query $Q = (w'_1, \ldots, w'_t)$ of the querier.

User-Request $(P_{\text{deBS}}, k_U, \text{pk}_{\text{KS}}, W = (w_1, \ldots, w_k))$:

- It returns $\perp$ if $H = 1_G$ or $e_2(H, \hat{g}) \neq e_2(g, \hat{H})$;
- For $i = 1$ to k, it computes $r_i \leftarrow F^1_{k_U}(w_i)$, and sets $R = (r_1, \ldots, r_k)$;
- $\text{Com} = (\text{Com}_1, \ldots, \text{Com}_k) = (g^{w_1} H^{r_1}, \ldots, g^{w_k} H^{r_k})$;
- It returns $(\xi = \text{Com}, \text{st} = (W, R))$, and sends ξ to the keyword server.

KeywordServer-Issue $(P_{\text{deBS}}, \text{sk}_{\text{KS}} = (\bar{h}, x, y, k_{\text{KS}}), \xi)$:

- For $i = 1$ to k, it computes $a'_i \leftarrow F^2_{k_{\text{KS}}}(\text{Com}_i)$, and sets $a' = (a'_1, \ldots, a'_k)$, and $\bar{\sigma} = \{\bar{\sigma}_i\}_{i=1}^k = \{(A'_i, B'_i, C'_i)\}_{i=1}^k$, where $\bar{\sigma}_i = (A'_i, B'_i, C'_i) = (g^{a'_i}, (g^x \text{Com}_i)^{\frac{a'_i}{y}}, H^{\frac{a'_i}{y}})$;
- It returns $\bar{\sigma}$ to the user.

User-Process $(P_{\text{deBS}}, k_U, \text{pk}_{\text{KS}}, \bar{\sigma}, \text{st} = (W, R))$:

- It returns $\perp$ if $\{A'_i\}_{i=1}^k = 1_G$ or $\{e_2(C'_i, \hat{Y}) \neq e_2(A'_i, \hat{H})\}_{i=1}^k$;
- It then computes $\{B'_i = B'_i C'^{-r_i}_i\}_{i=1}^k$;
- It returns $\perp$ if $\{e_2(B'_i, \hat{Y}) \neq e_2(A'_i, \hat{X}\hat{g}^{w_i})\}_{i=1}^k$;
- For $i = 1$ to k, it computes $a_i \leftarrow F^1_{k_U}(r_i)$, and sets $a = (a_1, \ldots, a_k)$.

Finally, it outputs $\sigma_W = \{\sigma_{w_i}\}_{i=1}^k$, where $\sigma_{w_i} = (A_i, B_i) = (A_i^{'a_i}, B_i^{'a_i})$. The user obtains the PKW $\text{pkw}_W = \{\text{pkw}_{w_i}\}_{i=1}^k = \{A_i || B_i\}_{i=1}^k$.

• SA-dPECKS$(P_{\text{PECKS}}, \text{pk}_{\text{SS}}, \text{pk}_{\text{QU}}, \text{pkw}_D)$: For the PKW $\text{pkw}_D = \{\text{pkw}_{w_i}\}_{i=1}^m$, an m-degree polynomial $f(X) = b_m X^m + b_{m-1} X^{m-1} + \cdots + b_1 X + b_0$ is constructed so that $\beta h(\text{pkw}_{w_1}), \ldots, \beta h(\text{pkw}_{w_m})$ are

m roots of the equation $f(X)-1=0$. This polynomial would accompany pkw_D as a default input to the algorithm which can be predefined by the querier for each document's PKW (in fact, the data owner and querier are the same person in most scenarios). Then, it selects values $\tau, \mu \xleftarrow{\$} \mathbb{Z}_P^*$, and computes $v_1=\gamma\cdot e_1(g_1,g_2)^{-\mu}$, $v_2=g_1^{\tau}$, $C_j=g_2^{\mu\cdot b_j}(0\leqslant j\leqslant m)$, where $\gamma=e_1(b_{\mathrm{SS}},a_{\mathrm{SS}})^{\tau}$. Finally, it sets PECKS ciphertext $\mathrm{CT}_{\mathrm{pkw}_D}=\{v_1,v_2,C_0,C_1,\ldots,C_m\}$.

• SA-dTrapdoor$(P_{\mathrm{PECKS}},\mathrm{pk}_{\mathrm{SS}},\mathrm{sk}_{\mathrm{QU}},\mathrm{pkw}_Q)$: For the PKW $\mathrm{pkw}_Q=\{\mathrm{pkw}_{w_r'}\}_{r=1}^{t}$, the algorithm selects $\varphi,\pi\xleftarrow{\$}\mathbb{Z}_P^*$, and lets $T_{Q,O_1}=\varphi$. Then it computes $T_{Q,O_2}=g_1^{\pi}$ and $T_{Q,j}=g_1^{t^{-1}\cdot T_{Q,O_1}\cdot\beta^j\cdot\sum_{r=1}^{t}h(\mathrm{pkw}_{w_r'})^j}\cdot(b_{\mathrm{SS}})^{\pi}$, where $0\leqslant j\leqslant m$. It returns the trapdoor $T_{\mathrm{pkw}_Q}=(T_{Q,O_1},T_{Q,O_2},T_{Q,0},T_{Q,1},\ldots,T_{Q,m})$ and sends T_{pkw_Q} to the SS.

• SA-dTest$(P_{\mathrm{PECKS}},\mathrm{sk}_{\mathrm{SS}},T_{\mathrm{pkw}_Q},\mathrm{CT}_{\mathrm{pkw}_D})$: First, the algorithm computes $\gamma=e_1(v_2,a_{\mathrm{SS}})^{\mathrm{sk}_{\mathrm{SS}}}=e_1(v_2,a_{\mathrm{SS}})^{\alpha}$, and computes $u=\prod_{j=0}^{m}e_1(T_{Q,j}/T_{Q,O_2}^{\alpha},C_j)$. Then, it checks if $v_1^{T_{Q,O_1}}\cdot u=\gamma^{T_{Q,O_1}}$. If so, output "True", and "False" otherwise.

6 Analysis of our scheme

We now demonstrate that our SA-SCF-PECKS scheme is correct and secure under our security model.

6.1 Correctness

Theorem 1. The correctness of $\Pi_{\text{SA-SCF-PECKS}}$ holds if the deBS scheme is deterministic.

Proof. Firstly, for the deBS scheme is deterministic, the algorithm SA-PKW$(P_{\mathrm{deBS}},k_U,\mathrm{pk}_{\mathrm{KS}},\mathrm{sk}_{\mathrm{KS}},W)$ in our scheme is also deterministic. Then we have that for any keyword set W,W', for a User's secret key k_U, if $W=W'$, $\sigma_W=\sigma_{W'}$, then $\mathrm{pkw}_W=\mathrm{pkw}_{W'}$. Next, we have $v_1^{T_{Q,O_1}}=e_1(b_{\mathrm{SS}},a_{\mathrm{SS}})^{\tau\varphi}\cdot e_1(g_1,g_2)^{-\mu\varphi}$. Based on equation $f(X)$,

$$\begin{aligned}u=\prod_{j=0}^{m}e_1\left(\frac{T_{Q,j}}{T_{Q,O_2}^{\alpha}},C_j\right)&=\prod_{j=0}^{m}e_1\left(g_1^{t^{-1}\cdot\varphi\cdot\beta^j\cdot\sum_{r=1}^{t}h(\mathrm{pkw}_{w_r'})^j},g_2^{\mu\cdot b_j}\right)\\&=e_1(g_1^{t^{-1}\cdot\varphi},g_2^{\mu})^{\sum_{j=0}^{m}b_j\cdot\beta^j\cdot\sum_{r=1}^{t}h(\mathrm{pkw}_{w_r'})^j}\\&=e_1(g_1,g_2)^{\mu\varphi}.\end{aligned}$$

Thus, if $v_1^{T_{Q,O_1}}\cdot u=e_1(b_{\mathrm{SS}},a_{\mathrm{SS}})^{\tau\varphi}=\gamma^{T_{Q,O_1}}$, we have $\mathrm{pkw}_Q\subseteq\mathrm{pkw}_D$, so $Q\subseteq D$. The algorithm SA-dTest will output "True"; otherwise, it will output "False".

6.2 Security

Theorem 2. $\Pi_{\text{SA-SCF-PECKS}}$ is secure in the standard model if the DDH problem is intractable and the underlying blind signature deBS has the property of blindness and one-more unforgeability.

The proof of Theorem 2 follows from Lemmas 3–6.

Lemma 3. $\Pi_{\text{SA-SCF-PECKS}}$ is IND-KGA secure in the standard model if the DDH problem is hard.

Proof. Let the winning advantage of a PPT adversary $\mathcal{A}_1$ in $\mathrm{Exp}_{\mathcal{A}(\mathrm{OA})}^{\text{IND-KGA}}(1^{\lambda})$ be $\mathrm{Adv}_{\mathcal{A}_1(\mathrm{OA})}^{\text{IND-KGA}}(1^{\lambda})$ and the advantage of a PPT adversary $\mathcal{B}_1$ successfully solving the DDH problem be $\mathrm{Adv}_{\mathcal{B}_1}^{\mathrm{DDH}}(1^{\lambda})$. We construct the algorithm $\mathcal{B}_1$ by taking $\mathcal{A}_1$ as a subroutine. Assume $\mathcal{B}_1$ has been given an instance $(g_1,g_1^{\delta_1},g_1^{\delta_2},Z)$ of the DDH problem. Then the experiment between $\mathcal{A}_1$ and $\mathcal{B}_1$ can be executed as follows.

• Setup. $\mathcal{B}_1$ chooses a value $a_{\mathrm{SS}}\xleftarrow{\$}G_2$, computes $b_{\mathrm{SS}}=g_1^{\delta_2}$, and sets $\mathrm{pk}_{\mathrm{SS}}=(a_{\mathrm{SS}},b_{\mathrm{SS}})$, $\mathrm{sk}_{\mathrm{SS}}=\delta_2$. Then, $\mathcal{B}_1$ selects a value $\beta\xleftarrow{\$}\mathbb{Z}_P^*$, sets private key $\mathrm{sk}_{\mathrm{QU}}=\beta$, and computes $\mathrm{pk}_{\mathrm{QU}}=g_1^{\beta}$. $\mathcal{B}_1$ continues to select values $\bar{h},x,y\xleftarrow{\$}\mathbb{Z}_P^*$, $k_{\mathrm{KS}}\xleftarrow{\$}\{0,1\}^{\lambda}$ and sets private key $\mathrm{sk}_{\mathrm{KS}}=(\bar{h},x,y,k_{\mathrm{KS}})$. Then it computes public key $\mathrm{pk}_{\mathrm{KS}}=(H,\hat{H},\hat{X},\hat{Y})=(g^{\bar{h}},\hat{g}^{\bar{h}},\hat{g}^{x},\hat{g}^{y})$. Finally, $\mathcal{B}_1$ sends $(\mathrm{pk}_{\mathrm{QU}},\mathrm{pk}_{\mathrm{SS}},\mathrm{pk}_{\mathrm{KS}})$ to $\mathcal{A}_1$.

• Query-I. $\mathcal{A}_1$ adaptively issues $\mathcal{B}_1$ the trapdoor of the conjunctive keyword query $Q=\{W_1',\ldots,W_t'\}$ $(t\leqslant m)$. $\mathcal{B}_1$ first runs the SA-PKW$(P_{\mathrm{deBS}},k_U,\mathrm{pk}_{\mathrm{KS}},\mathrm{sk}_{\mathrm{KS}},Q)$ to generate pkw_Q. Then it chooses

values $\varphi, \pi \xleftarrow{\$} \mathbb{Z}_P^*$, and sets $T_{Q,O_1} = \varphi$. For $0 \leqslant j \leqslant m$, it computes $T_{Q,O_2} = g_1^{\pi}$ and $T_{Q,j} = g_1^{t^{-1} \cdot T_{Q,O_1} \cdot \beta^j \cdot \sum_{r=1}^{t} h(\mathrm{pkw}_{w_r'})^j} \cdot (b_{\mathrm{SS}})^{\pi}$, and responds the trapdoor $T_{\mathrm{pkw}_Q} = (T_{Q,O_1}, T_{Q,O_2}, T_{Q,0}, T_{Q,1}, \ldots, T_{Q,m})$.

• Challenge. $\mathcal{A}_1$ chooses two keyword sets Q_0, Q_1 which have not been queried in the Query-I stage and sends them to $\mathcal{B}_1$. $\mathcal{B}_1$ chooses $b \in \{0,1\}$, runs SA-PKW$(P_{\mathrm{deBS}}, k_U, \mathrm{pk}_{\mathrm{KS}}, \mathrm{sk}_{\mathrm{KS}}, Q_b)$ to generate pkw_{Q_b}. Then $\mathcal{B}_1$ selects an element $\varphi^* \xleftarrow{\$} \mathbb{Z}_P^*$ and lets $T^*_{Q_b,O_1} = \varphi^*$, $T^*_{Q_b,O_2} = g_1^{\delta_1}$, $T^*_{Q_b,j} = g_1^{t^{-1} \cdot T^*_{Q_b,O_1} \cdot \beta^j \cdot \sum_{r=1}^{t} h(\mathrm{pkw}_{w_r^*})^j} \cdot Z$. Finally, $\mathcal{B}_1$ sends $T^* = (T^*_{Q_b,O_1}, T^*_{Q_b,O_2}, \{T^*_{Q_b,j}\}_{j=1}^m)$ to $\mathcal{A}_1$. If T^* is a valid trapdoor for the conjunctive keyword query Q_b, then $Z = g_1^{\delta_1\delta_2}$, so $T^*_{Q_b,O_2} = g_1^{\pi^*}, T^*_{Q_b,j} = g_1^{t^{-1} \cdot T^*_{Q_b,O_1} \cdot \beta^j \cdot \sum_{r=1}^{t} h(\mathrm{pkw}_{w_r^*})^j} \cdot (b_{\mathrm{SS}})^{\pi^*}$, where $\pi^* = \delta_1$.

• Query-II. $\mathcal{A}_1$ repeats the process as in the Query-I stage, the restriction is Q_0, Q_1 cannot be queried.

• Output. $\mathcal{A}_1$ outputs its guess b', if $b = b'$, $\mathcal{B}_1$ returns 1 meaning $Z = g_1^{\delta_1\delta_2}$; else, it outputs 0 meaning Z is a random element in group G_1.

From the experiment above, we have $\mathrm{Adv}^{\mathrm{DDH}}_{G_1,\mathcal{B}_1}(1^\lambda) \geqslant \mathrm{Adv}^{\text{IND-KGA}}_{\mathcal{A}_1(\mathrm{OA})}(1^\lambda)$. As the DDH problem is intractable, that is $\mathrm{Adv}^{\mathrm{DDH}}_{G_1,\mathcal{B}_1}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$, then $\mathrm{Adv}^{\text{IND-KGA}}_{\mathcal{A}_1(\mathrm{OA})}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$.

Lemma 4. $\Pi_{\text{SA-SCF-PECKS}}$ is IND-CKGA secure in the standard model if the underlying blind signature deBS has the property of one-more unforgeability.

Proof. Assume $\mathcal{A}_2$ is a PPT adversary in $\mathrm{Exp}^{\text{IND-CKGA}}_{\mathcal{A}(\mathrm{SS})}(1^\lambda)$ with the winning advantage $\mathrm{Adv}^{\text{IND-CKGA}}_{\mathcal{A}_2(\mathrm{SS})}(1^\lambda)$, then a PPT adversary $\mathcal{B}_2$ can be built to break the one-more unforgeability of deBS with the winning advantage $\mathrm{Adv}^{\mathrm{OMU}}_{\mathrm{deBS},\mathcal{B}_2}(1^\lambda)$, who also simulates the challenger in $\mathrm{Exp}^{\text{IND-CKGA}}_{\mathcal{A}(\mathrm{SS})}(1^\lambda)$. Let the challenger in the one-more unforgeability security breaking experiment be $\mathcal{C}_2$.

• Setup. A public key $\mathrm{pk}_{\mathrm{KS}}$ is generated and sent to $\mathcal{B}_2$ by $\mathcal{C}_2$. $\mathcal{B}_2$ then runs GlobalSetup(1^λ) to generate the global parameters $\mathcal{GP} = (P_{\mathrm{deBS}}, P_{\mathrm{PECKS}})$, calls the algorithms SA-KeyGen$_U$, SA-KeyGen$_{\mathrm{QU}}$ and SA-KeyGen$_{\mathrm{SS}}$ to generate the corresponding keys, and sends $(\mathrm{pk}_{\mathrm{QU}}, \mathrm{pk}_{\mathrm{KS}}, \mathrm{sk}_{\mathrm{SS}}, \mathrm{pk}_{\mathrm{SS}})$ to $\mathcal{A}_2$.

• Query-I. $\mathcal{A}_2$ can adaptively ask $\mathcal{B}_2$ for the PECKS ciphertext (or the trapdoor) of its query keyword set $W_{q_{\mathrm{I}}} = \{w_1, w_2, \ldots, w_k\}$. For the query, $\mathcal{B}_2$ first interacts with $\mathcal{C}_2$ to get the signature $\sigma_{W_{q_{\mathrm{I}}}} = \{\sigma_{w_i}\}_{i=1}^k = \{(A_i, B_i)\}_{i=1}^k$, and sets $\mathrm{pkw}_{W_{q_{\mathrm{I}}}} = \{\mathrm{pkw}_{w_i}\}_{i=1}^k = \{A_i || B_i\}_{i=1}^k$. Then it runs SA-dPECKS $(P_{\mathrm{PECKS}}, \mathrm{pk}_{\mathrm{SS}}, \mathrm{pk}_{\mathrm{QU}}, \mathrm{pkw}_{W_{q_{\mathrm{I}}}})$ to generate $\mathrm{CT}_{\mathrm{pkw}_{W_{q_{\mathrm{I}}}}}$ (or runs SA-dTrapdoor$(P_{\mathrm{PECKS}}, \mathrm{pk}_{\mathrm{SS}}, \mathrm{sk}_{\mathrm{QU}}, \mathrm{pkw}_{W_{q_{\mathrm{I}}}})$ to generate $T_{\mathrm{pkw}_{W_{q_{\mathrm{I}}}}}$), and responds the result to $\mathcal{A}_2$. Assume $\mathcal{A}_2$ has queried q_{I_k} distinct keyword sets in this stage.

• Challenge. The challenge keyword sets W_0^*, W_1^* are chosen by $\mathcal{A}_2$ with the constraint $W_b^* \cap (W_1 \cup \cdots \cup W_{q_{\mathrm{I}_k}}) = \phi$, where $b \in \{0,1\}$, $W_b^* = \{w_{b_1}^*, \ldots, w_{b_k}^*\}$, and sent to $\mathcal{B}_2$. $\mathcal{B}_2$ chooses $b \in \{0,1\}$ randomly. Instead of querying W_b^* to $\mathcal{C}_2$ for the signature $\sigma_{W_b^*}$, $\mathcal{B}_2$ just randomly picks $r = \{(r_{1L}, r_{1R}), \ldots, (r_{kL}, r_{kR})\}$, where $r_{iL}, r_{iR} \xleftarrow{\$} G, i \in [1,k]$, and sets $\mathrm{pkw}_{W_b^*} = \{\mathrm{pkw}_{w_{b_i}^*}\}_{i=1}^k = \{r_{iL} || r_{iR}\}_{i=1}^k$. Finally, it generates the corresponding ciphertext and trapdoor (CT^*, T^*) as before, and sends (CT^*, T^*) to $\mathcal{A}_2$.

• Query-II. $\mathcal{A}_2$ again asks for the ciphertexts and trapdoors of any keyword set $W_{q_{\mathrm{II}}}$. $\mathcal{B}_2$ simulates as in the Query-I stage. Assuming that $\mathcal{A}_2$ has queried q_{II_k} distinct keyword sets $\{W_1, W_2, \ldots, W_{q_{\mathrm{II}_k}}\}$ in Query-II stage, we require that any keyword set $W_{q_{\mathrm{II}}} \cap (W_1^* \cup W_0^*) = \phi$.

• Output. $\mathcal{A}_2$ outputs its guess b' on b. If $b = b'$, then $\mathrm{pkw}_{W_b^*}$ is a valid PKW of W_b^*, $\mathcal{B}_2$ outputs $q_{\mathrm{I}_k} + q_{\mathrm{II}_k}$ pair $\{W_i, \sigma_{W_i}\}_{i=1}^{q_{\mathrm{I}_k}+q_{\mathrm{II}_k}}$ and another pair $\{W_b^*, r\}$ which meet the following three requirements: (1) $\forall i, j \in [1, q_{\mathrm{I}_k} + q_{\mathrm{II}_k}]$ and $i \neq j$, $W_i \neq W_j$, (2) $W_b^* \cap (W_1 \cup \cdots \cup W_{q_{\mathrm{I}_k}+q_{\mathrm{II}_k}}) = \phi$, and (3) $\forall i \in [1, q_{\mathrm{I}_k} + q_{\mathrm{II}_k}]$, σ_{W_i} and $\sigma_{W_b^*} = r$ are all valid signatures. Then, $\mathcal{B}_2$ directly outputs these $q_{\mathrm{I}_k} + q_{\mathrm{II}_k} + 1$ valid {keyword sets, signature} pairs as its valid forgeries; else, $\mathcal{B}_2$ returns $\perp$.

From the experiment, we have $\mathrm{Adv}^{\mathrm{OMU}}_{\mathrm{deBS},\mathcal{B}_2}(1^\lambda) \geqslant \mathrm{Adv}^{\text{IND-CKGA}}_{\mathcal{A}_2(\mathrm{SS})}(1^\lambda)$. As deBS has the property of one-more unforgeability, i.e., $\mathrm{Adv}^{\mathrm{OMU}}_{\mathrm{deBS},\mathcal{B}_2}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$, then $\mathrm{Adv}^{\text{IND-CKGA}}_{\mathcal{A}_2(\mathrm{SS})}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$.

Lemma 5. $\Pi_{\text{SA-SCF-PECKS}}$ is IND-CKA secure in the standard model if the underlying blind signature deBS has the property of blindness.

Proof. Suppose $\mathcal{A}_3$ is a PPT adversary in $\mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}(\mathrm{KS})}(1^\lambda)$ with the winning advantage $\mathrm{Adv}^{\text{IND-CKA}}_{\mathcal{A}_3(\mathrm{KS})}(1^\lambda)$, then we can invoke $\mathcal{A}_3$ to construct a PPT adversary $\mathcal{B}_3$ (i.e., the adversarial signer who is curious about

the information of User's private input (keyword set)) to break the blindness of deBS with the winning advantage $\mathrm{Adv}^{\mathrm{Blindness}}_{\mathrm{deBS},\mathcal{B}_3}(1^\lambda)$, who also simulates the challenger in $\mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}(\mathrm{KS})}(1^\lambda)$. Let the challenger in the blindness breaking experiment be $\mathcal{C}_3$.

• Setup. $\mathcal{C}_3$ sends $(\mathrm{pk}_{\mathrm{KS}}, \mathrm{sk}_{\mathrm{KS}})$ to $\mathcal{B}_3$ who then forwards this key pair to $\mathcal{A}_3$. Upon receiving the challenge keyword sets (W_0, W_1) from $\mathcal{A}_3$, $\mathcal{B}_3$ passes (W_0, W_1) on to $\mathcal{C}_3$ as the challenge messages.

• Challenge. Two executions are simulated. During the first execution of deBS, once $\mathcal{C}_3$ starts executing deBS, $\mathcal{B}_3$ begins interacting with $\mathcal{A}_3$ to run the keyword preprocessing protocol and delivers the message between $\mathcal{A}_3$ and $\mathcal{C}_3$. In the second execution, $\mathcal{B}_3$ engages with $\mathcal{C}_3$ by honestly utilizing $(\mathrm{pk}_{\mathrm{KS}}, \mathrm{sk}_{\mathrm{KS}})$.

• Output. $\mathcal{A}_3$ returns its guess b', then $\mathcal{B}_3$ returns b' as its guess to $\mathcal{C}_3$.

From the view of $\mathcal{A}_3$, the IND-CKA experiment $\mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}(\mathrm{KS})}(1^\lambda)$ is clearly indistinguishable from this simulation. Thus, we have $\mathrm{Adv}^{\mathrm{Blindness}}_{\mathrm{deBS},\mathcal{B}_3}(1^\lambda) \geqslant \mathrm{Adv}^{\text{IND-CKA}}_{\mathcal{A}_3(\mathrm{KS})}(1^\lambda)$. As the underlying scheme deBS has the property of blindness, i.e., $\mathrm{Adv}^{\mathrm{Blindness}}_{\mathrm{deBS},\mathcal{B}_3}(1^\lambda) \leqslant \mathrm{negl}\,(1^\lambda)$, then $\mathrm{Adv}^{\text{IND-CKA}}_{\mathcal{A}_3(\mathrm{KS})}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$.

Lemma 6. $\Pi_{\text{SA-SCF-PECKS}}$ is OMU-CKA secure in the standard model if the underlying blind signature deBS has the property of one-more unforgeability.

Proof. Assume $\mathcal{A}_4$ is a PPT adversary in $\mathrm{Exp}^{\text{OMU-CKA}}_{\mathcal{A}(U)}(1^\lambda)$ with the winning advantage $\mathrm{Adv}^{\text{OMU-CKA}}_{\mathcal{A}_4(U)}(1^\lambda)$, then a PPT adversary $\mathcal{B}_4$ (i.e., the adversarial user who attempts to generate a new valid signature based on the signatures obtained previously) can be constructed to break the one-more unforgeability of deBS with the winning advantage $\mathrm{Adv}^{\mathrm{OMU}}_{\mathrm{deBS},\mathcal{B}_4}(1^\lambda)$, who also simulates the challenger in $\mathrm{Exp}^{\text{OMU-CKA}}_{\mathcal{A}(U)}(1^\lambda)$. Let the challenger in the one-more unforgeability breaking experiment be $\mathcal{C}_4$.

• Setup. $\mathcal{B}_4$ receives the secret key k_U and the public key $\mathrm{pk}_{\mathrm{KS}}$ from $\mathcal{C}_4$, and forwards keys to $\mathcal{A}_4$.

• PKW-Query. Once $\mathcal{A}_4$ issues a query about the keyword set W, $\mathcal{B}_4$ queries W to $\mathcal{C}_4$ to get the signature $\sigma_W = \{\sigma_{w_i}\}_{i=1}^k = \{(A_i, B_i)\}_{i=1}^k$, and sets $\mathrm{pkw}_W = \{\mathrm{pkw}_{w_i}\}_{i=1}^k = \{A_i || B_i\}_{i=1}^k$. Finally, $\mathcal{B}_4$ returns the corresponding PKW pkw_W of W to $\mathcal{A}_4$.

• Output. If $\mathcal{A}_4$ outputs q_k pairs $\{W_c, \mathrm{pkw}_{W_c}\}_{c=1}^{q_k}$ and another pair $\{\overline{W}, \mathrm{pkw}_{\overline{W}}\}$ that satisfy the conditions: (1) $W_c \neq W_n$, for any $c, n \in [1, q_k]$ where $c \neq n$, (2) $\overline{W} \cap (W_1 \cup \cdots \cup W_c \cup \cdots \cup W_{q_k}) = \phi$, and (3) pkw_{W_c} for any $c \in [1, q_k]$ and $\mathrm{pkw}_{\overline{W}}$ are all valid PKWs, where q_k is the number of the PKW-Queries. Then, $\mathcal{B}_4$ parses each pkw_{W_c} and $\mathrm{pkw}_{\overline{W}}$ as the signatures $\{\{(A_{c,i}, B_{c,i})\}_{i=1}^k\}_{c=1}^{q_k}$, $\{(\bar{A}_i, \bar{B}_i)\}_{i=1}^k$. Finally $\mathcal{B}_4$ outputs these $q_k + 1$ valid {keyword sets, signature} pairs as its valid forgeries. Else, $\mathcal{B}_4$ returns $\perp$.

From the experiment, $\mathcal{B}_4$'s simulation is indistinguishable from $\mathrm{Exp}^{\text{OMU-CKA}}_{\mathcal{A}(U)}(1^\lambda)$ in the perspective of $\mathcal{A}_4$ such that $\mathrm{Adv}^{\mathrm{OMU}}_{\mathrm{deBS},\mathcal{B}_4}(1^\lambda) \geqslant \mathrm{Adv}^{\text{OMU-CKA}}_{\mathcal{A}_4(U)}(1^\lambda)$. As the underlying scheme deBS has the property of one-more unforgeability, i.e., $\mathrm{Adv}^{\mathrm{OMU}}_{\mathrm{deBS},\mathcal{B}_4}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$, then $\mathrm{Adv}^{\text{OMU-CKA}}_{\mathcal{A}_4(U)}(1^\lambda) \leqslant \mathrm{negl}(1^\lambda)$.

Discussions on (offline) OKGA and IKGA. In our SA-SCF-PECKS scheme, the user must perform mutual authentication with KS and interact with it to get the PKW before computing the ciphertext. Due to the one-more unforgeability of the underlying scheme deBS, the advantage of adversary $\mathcal{A}$ in forging a valid keyword set/signature pair based on the previously obtained PKWs is negligible, i.e., the adversary cannot generate the ciphertext of his guessed keyword set by himself. We note that $\mathcal{A}$ cannot even produce a valid PKW in an online manner since the keyword set preprocessing requires the User (i.e., data owner and querier) to share the same secret key. Therefore, $\mathcal{A}$ is unable to attack our SA-SCF-PECKS scheme successfully by performing such OKGA and IKGA.

7 Implementation and performance

We theoretically compare the performance of our scheme with some related schemes (in Table 2 [6, 9, 11, 13, 14]). Compared with variable keyword field schemes, our scheme has relatively lower computational cost.

We implement our proposed SA-SCF-PECKS scheme and the PECSK scheme in ZZ11 [13], which is an efficient variable keyword field scheme and has application scenarios similar to that for our scheme. Our experiments are conducted on a laptop with a 3.40 GHz ×8 Intel Core i7-3770 CPU, 24 GB memory and Ubuntu 16.04 LTS 64-bit operating system. We use the C language and PBC Library. The experimental

Table 2 Performance comparison of various PECKS schemes[a)]

	|pk|	|sk|	|CT|	|T|	Encryption	Trapdoor	Test	Type
GSW04 [9]	–	$\|\lambda\|$	$(m+1)\|G_1\|_b$	$(n+1)\|\mathbb{Z}_P^*\|_b$	$(m+1)E$	$(n)E+(n)H$	$2E+H$	Fixed keyword field
PKL05 [6] (Scheme 1)	$3\|G_1\|_b$	$2\|\mathbb{Z}_P^*\|_b$	$2\|G_1\|_b+(m)\|G_T\|_b$	$\|G_1\|_b+\|\mathbb{Z}_P^*\|_b$	$(m)P+(m)H+(m+2)E$	$(t)H+E$	P	Fixed keyword field
CH09 [11]	$2\|G_1\|_b+\|G_T\|_b$	$\|\mathbb{Z}_P^*\|_b$	$(m+1)\|G_1\|_b$	$3\|G_T\|_b$	$(m)H+(m+1)E$	$(t)H+2E$	$2P$	Fixed keyword field
ZZ11 [13]	$(m+3)\|G_1\|_b+2\|G_T\|_b$	$\|\mathbb{Z}_P^*\|_b$	$(m+1)\|G_1\|_b+(m+2)\|G_T\|_b+\|\mathbb{Z}_P^*\|_b$	$(m+2)\|G_1\|_b+\|G_T\|_b$	$(m+1)H+P+(2m+4)E$	$(t)H+(2m+5)E$	$(2m+3)P$	Variable keyword field
MML17 [14]	CSP: $2\|G_1\|_b$ Data owner: $\|G_1\|_b$ Querier: $\|G_1\|_b$ KS: $4\|G_1\|_b$	CSP: $\|\mathbb{Z}_P^*\|_b$ Data owner: $\|\mathbb{Z}_P^*\|_b$ Querier: $\|\mathbb{Z}_P^*\|_b$ KS: $3\|\mathbb{Z}_P^*\|_b+\|\lambda\|$	$(m+3)\|G_1\|_b+\|G_T\|_b$	$(m+2)\|G_1\|_b+\|\mathbb{Z}_P^*\|_b$	$2P+(m+6)E+(m+2)H$	$(t)H+(m+2)E$	$4E+(m+2)P$	Variable keyword field
Our scheme	SS: $2\|G_1\|_b$ Querier: $\|G_1\|_b$	SS: $\|\mathbb{Z}_P^*\|_b$ Querier: $\|\mathbb{Z}_P^*\|_b$	$(m+2)\|G_1\|_b+\|G_T\|_b$	$(m+2)\|G_1\|_b+\|\mathbb{Z}_P^*\|_b$	$2P+(m+4)E+(m)H$	$(t)H+(m+2)E$	$4E+(m+2)P$	Variable keyword field

a) Assume $|G_1|_b = |G_2|_b$, the symmetric bilinear pairing is $e' : G_1 \times G_1 \to G_T$ and the Type-III bilinear pairing is $e_1 : G_1 \times G_2 \to G_T$. E, P, and H denote modular exponentiation, pairing and hash operation, respectively. m (respectively, t): the number of different keywords in the document (respectively, the query). We consider the case in MML17 [14] where there are only one data owner, one querier and the data owner just deals with one document (EHR).

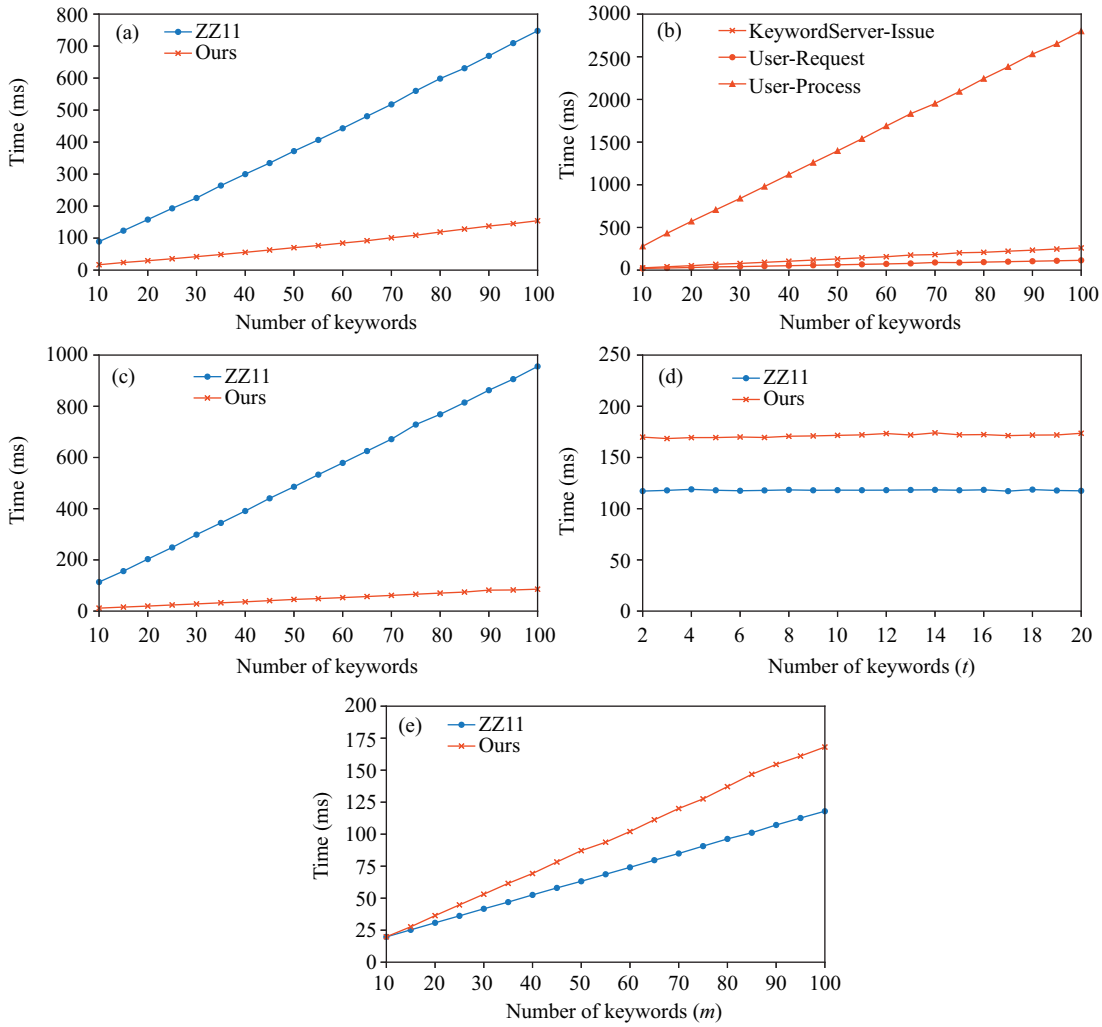

Figure 3 (Color online) Running time of (a) PECKS, (b) PKW, (c) Test, (d) Trapdoor (fixed $m = 100$), (e) Trapdoor (fixed $t = 5$).

results are shown in Figures 3(a)–(e).

Figure 3(a) shows the running time of the encryption algorithm (SA-dPECKS in our scheme and PECSK in ZZ11) executed by the data owner. The encryption time of PECSK in ZZ11 and our scheme increases with the number of keywords m. Our SA-SCF-PECKS scheme is much more efficient than the PECSK scheme.

Theoretical observation reveals that the efficiency of the algorithm Trapdoor is related to two parameters: the number of keywords (m) in the documents and the number of keywords (t) in the query. We

conducted two experiments: one when $m = 100$ (in Figure 3(d)) and the other when $t = 5$ (as shown in Figure 3(e)). However, Figure 3(d) shows the running time of both schemes is nearly independent of t when $m = 100$. Because the operations associated with parameter t are hash, add, and multiplication operations. These operations are much faster than the modular exponentiation and pairing operations. Therefore, as the number of keywords (t) increases, these operations have little impact on the execution time of the algorithm Trapdoor and can be omitted. Figure 3(e) shows that the running time of algorithm Trapdoor in both schemes increases with the number of keywords m. We observe the running time of algorithm Trapdoor in our scheme is slightly longer than that of the corresponding algorithm in ZZ11 which differs from the results in Table 2. We attribute this variance to our omission of less time-consuming operations, such as add and multiplication, in the theoretical analysis. As shown in Figure 3(c), the algorithm Test executed by the SS in our scheme performs very well. In other words, our scheme can achieve quick search responses, which is crucial for searchable encryption schemes.

The algorithm PKW in our scheme is a three-phase protocol between the User and the KS. As shown in Figure 3(b), the running time of the whole algorithm is proportional to the number of keywords m. The main overhead occurs in the User-Process phase because the pairing operation needed in this phase is time-consuming. However, the scheme is also efficient, and its running time is less than 3 s when the number of keywords is 100. Algorithm PKW has little impact on the search efficiency of the SS. It is worth stressing that this keyword preprocessing protocol can be conducted ahead of time for later use. On the basis of the experimental results, we conclude our SA-SCF-PECKS scheme is efficient and the resistance against full KGA has little influence on its efficiency.

8 Conclusion

In this paper, based on the in-depth discussion of the OKGA and IKGA, we propose a new framework called SA-SCF-PECKS for secure data outsourcing with enhanced privacy protection, which can resist full KGA. Under this framework, we construct the first PECKS scheme that is secure against both OKGA and IKGA in the standard model. No secure channel between the SS and the querier is needed. A detailed proof of security and theoretical and practical performance analyses are provided for our scheme. Our SA-SCF-PECKS scheme shows good performance in terms of security and efficiency. Moreover, it can be integrated to the cloud platform, such as ownCloud, for further applications.

Acknowledgements This work was supported in part by National Natural Science Foundation of China (Grant Nos. 61632020, 61472416, 61772520, 61802392, 61972094), Key Research Project of Zhejiang Province (Grant No. 2017C01062), and Beijing Municipal Science and Technology Project (Grant Nos. Z191100007119007, Z191100007119002). The authors thank Yang TAO for facilitating many helpful discussions.

Supporting information Appendixes A and B. The supporting information is available online at info.scichina.com and link.springer.com. The supporting materials are published as submitted, without typesetting or editing. The responsibility for scientific accuracy and content remains entirely with the authors.

References

1 Gentry C, Boneh D. A Fully Homomorphic Encryption Scheme. Stanford: Stanford University, 2009
2 Goldreich O. Towards a theory of software protection and simulation by oblivious RAMs. In: Proceedings of the 19th Annual ACM Symposium on Theory of Computing. New York: ACM, 1987. 182–194
3 Naveed M. The fallacy of composition of oblivious RAM and searchable encryption. IACR Cryptol ePrint Archive, 2015, 2015: 668
4 Boneh D, Di Crescenzo G, Ostrovsky R, et al. Public key encryption with keyword search. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2004. 506–522
5 Baek J, Safavi-Naini R, Susilo W. Public key encryption with keyword search revisited. In: Proceedings of International Conference on Computational Science and Its Applications. Berlin: Springer, 2008. 1249–1259
6 Park D J, Kim K, Lee P J. Public key encryption with conjunctive field keyword search. In: Proceedings of International Workshop on Information Security Applications. Berlin: Springer, 2004. 73–86
7 Canetti R, Goldreich O, Halevi S. The random oracle methodology, revisited. J ACM, 2004, 51: 557–594

8 Rhee H S, Park J H, Susilo W, et al. Improved searchable public key encryption with designated tester. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, 2009. 376–379
9 Golle P, Staddon J, Waters B. Secure conjunctive keyword search over encrypted data. In: Proceedings of International Conference on Applied Cryptography and Network Security. Berlin: Springer, 2004. 31–45
10 Hwang Y H, Lee P J. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Proceedings of International Conference on Pairing-based Cryptography. Berlin: Springer, 2007. 2–22
11 Chen Y C, Horng G. Timestamped conjunctive keyword-searchable public key encryption. In: Proceedings of the 4th International Conference on Innovative Computing, Information and Control (ICICIC). New York: IEEE, 2009. 729–732
12 Ryu E K, Takagi T. Efficient conjunctive keyword-searchable encryptio. In: Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07). New York: IEEE, 2007. 409–414
13 Zhang B, Zhang F G. An efficient public key encryption with conjunctive-subset keywords search. J Netw Comput Appl, 2011, 34: 262–267
14 Miao Y B, Ma J F, Liu X M, et al. VCKSM: verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. Pervasive Mobile Comput, 2017, 40: 205–219
15 Byun J W, Rhee H S, Park H A, et al. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Proceedings of Workshop on Secure Data Management. Berlin: Springer, 2006. 75–83
16 Fang L M, Susilo W, Ge C P, et al. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf Sci, 2013, 238: 221–241
17 Xu P, Jin H, Wu Q H, et al. Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack. IEEE Trans Comput, 2013, 62: 2266–2277
18 Guo L, Yau W C. Efficient secure-channel free public key encryption with keyword search for EMRs in cloud storage. J Med Syst, 2015, 39: 11
19 Wang C H, Tu T Y. Keyword search encryption scheme resistant against keyword-guessing attack by the untrusted server. J Shanghai Jiaotong Univ (Sci), 2014, 19: 440–442
20 Chen R M, Mu Y, Yang G M, et al. Dual-server public-key encryption with keyword search for secure cloud storage. IEEE Trans Inform Forensic Secur, 2015, 11: 789–798
21 Huang Q, Li H B. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf Sci, 2017, 403: 1–14
22 Jiang P, Mu Y, Guo F C, et al. Private keyword-search for database systems against insider attacks. J Comput Sci Technol, 2017, 32: 599–617
23 Sun L X, Xu C X, Zhang M W, et al. Secure searchable public key encryption against insider keyword guessing attacks from indistinguishability obfuscation. Sci China Inf Sci, 2018, 61: 038106
24 Hwang M S, Hsu S T, Lee C C. A new public key encryption with conjunctive field keyword search scheme. Inform Tech Control, 2014, 43: 3
25 Lu Y, Wang G, Li J G. On security of a secure channel free public key encryption with conjunctive field keyword search scheme. Inform Tech Control, 2018, 47: 56–62
26 Zhao Z Y, Wang J H. Novel multi-user conjunctive keyword search against keyword guessing attacks under simple assumptions. KSII Trans Internet Inform Syst, 2017, 11: 3699–3719
27 Yang Y, Ma M D. Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. IEEE Trans Inform Forensic Secur, 2016, 11: 746–759
28 Fang L, Susilo W, Ge C, et al. A secure channel free public key encryption with keyword search scheme without random oracle. In: Proceedings of International Conference on Cryptology and Network Security. Berlin: Springer, 2009. 248–258
29 Chen R M, Mu Y, Yang G M, et al. Server-aided public key encryption with keyword search. IEEE Trans Inform Forensic Secur, 2016, 11: 2833–2842
30 Baek J, Safavi-Naini R, Susilo W. On the integration of public key data encryption and public key encryption with keyword search. In: Proceedings of International Conference on Information Security. Berlin: Springer, 2006. 217–232
31 Ghadafi E. Efficient round-optimal blind signatures in the standard model. In: Proceedings of International Conference on Financial Cryptography and Data Security. Berlin: Springer, 2017. 455–473