Secure Searchable Public-Key Encryption for Cloud Storage

Run Xie[1(✉)], Changlian He[1], Yu He[2], Chunxiang Xu[2], and Kun Liu[3]

[1] Yibin University, Yibin, China
xrryun@126.com
[2] School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
[3] Department of Computer Science, Guangzhou University, Guangzhou, China

Abstract. With networking became prevalent, the amount of data to be stored and managed on networked servers rapidly increases. Meanwhile, with the improvement of awareness of data privacy, the user's sensitive data is usually encrypted before uploading them to the cloud server. The searchable public-key encryption provides an efficient mechanism to achieve data retrieval in encrypted storage. Therefore, it is a critical technique on promoting secure and efficient cloud storage. Unfortunately, only few the existing schemes are secure to resist outside keyword guessing attacks. In this paper, we propose two efficient searchable public-key encryption schemes with a designated tester (dPEKS). One is a basic dPEKS, where the dPEKS ciphertext indistinguishability is proved without the random oracle. Meanwhile, the basic scheme is secure to resist the outside KGA since it satisfies the property of trapdoor indistinguishability. Comparing with the existing dPEKS schemes which use expensive pairing computation, our scheme is more efficient since we only need multi-exponentiation. Another is an enhanced dPEKS scheme. With the sender's identity is kept secret from server, this scheme can provide stronger security.

Keywords: Searchable encryption · Trapdoor indistinguishability · Keywords guessing attacks · Cloud storage · Security analysis

1 Introduction

With ubiquitous network, the cloud storage offers great convenience to users. More and more users enjoy the benefits of cloud storage services by outsourcing their data into the cloud server. To protect data privacy, a user has to encrypt the sensitive data before uploading them into the server. However, this incurs a new problem that the network server cannot perform searches over encrypted data. When users want to retrieve the encrypted data, he has two straight options: downloading the entire encrypted data or sending his private keys to the cloud server. Obviously, the first approach requires high consumption of bandwidth and the second approach deviates original intention (namely protect data privacy).

J. Wan et al. (Eds.): CloudComp 2016, SPNCE 2016, LNICST 197, pp. 184–195, 2018.
https://doi.org/10.1007/978-3-319-69605-8_17

In 2000, Song et al. first introduced the concept of searchable encryption [1]. The searchable encryption allows the network server to search over encrypted data without decryption. It does not leak any information about the data and query. Therefore, searchable encryption is a critical technique promoting efficient and secure cloud storage. The searchable encryption has been developed into two different types. The first type is the symmetric searchable encryption (SSE in short) which requires that a sender is securely granted a secret key from the intended receiver. It suffers from risks of key leakage in management and distribution [1]. The second type is the searchable public-key encryption with keyword search (PEKS in short), which allows any one seeing the receiver's public key to encrypt documents.

The PEKS provides an efficient mechanism to achieve data retrieval in encrypted storage. In a PEKS scheme, the sender generates the searchable ciphertext of keywords with receiver's public key and stores it to server. To retrieval the encrypted data associated with a given keyword, the receiver creates a search request (trapdoor) with the keyword and his private key. Receiving a trapdoor, the cloud server can perform a test whether some encrypted data matches the trapdoor and returns corresponding encrypted data to receiver.

In 2004, Boneh et al. proposed the first searchable public-key encryption with keyword search scheme [2]. Their scheme requires constructing the secure transport channel to protect trapdoors. Since building a secure channel is usually expensive, this requirement limits applications of the searchable public-key encryption scheme.

To overcome this obstacle, in 2008, Baek et al. proposed secure Channel Free Public Key Encryption with Keyword Search scheme [3](SCF-PEKS in short), which removes the secure channel requirement. Nevertheless, Yau et al. showed that this scheme is insecure [4] for the following reason. With outside keyword-guessing attacks (outside KGA), an outside adversary can reveal encrypted keywords if he obtains a trapdoor in channel.

Hereafter, in [15], the searchable public key encryption with keyword search scheme with a designated tester (dPEKS in short) is proposed. In this scheme, only a designated server can test whether given trapdoor matches the ciphertext.

Until now, most of the dPEKS schemes pay more attention to improving the security against this attacks [5–11]. Only a few schemes [12–15] can effectively resist outside KGA.

In addition, the KGA launched by a server is called inside KGA. Since the correct requirement of scheme and small keyword space, it is impossible to construct a searchable public-key encryption(dPEKS or PEKS) scheme secure against inside KGA under the original framework [2]. Very recently, based on a new framework, Peng et. al. proposed a online/offline ciphertext retrieval scheme [16] is secure against inside KGA.

In this paper, based on the IBE [17], we propose two efficient dPEKS schemes, namely a basic dPEKS scheme (BdPEKS) and an enhanced dPEKS (EdPEKS). For the basic scheme, we prove that our construction satisfies ciphertext indistinguishability under q-ABDHE assumption. Meanwhile, we prove that it satisfies trapdoor indistinguishability. Therefore, our BdPEKS scheme is secure against outside keyword guessing attacks. Comparing with the existing dPEKS schemes which use expensive pairing computation, our basic scheme is more efficient since we only need multi-exponentiation. For our enhanced dPEKS scheme (EdPEKS), we analysis its security. In EdPEKS, if a server wants to launch the KGA, it must guess both the sender's identity and keywords. Therefore, the EdPEKS scheme has stronger security to resist the inside KGA. Lastly, we show a comparison between the other PEKS (dPEKS) schemes and our schemes in terms of functionalities and performances.

2 Preliminaries

In this section, we review the construction of dPEKS, which is defined in [15]. Meanwhile, we also describe the definition of dPEKS ciphertexts indistinguishability and trapdoor indistinguishability with game between the adversary $\mathcal{F}$ and the challenger $\mathcal{G}$. Here, the dPEKS ciphertext is an encrypted list of keywords.

2.1 Definition of dPEKS and Security Model

2.1.1 Definition of dPEKS

As stated in the previous section, the dPEKS is a mechanism which can achieve efficient ciphertext retrieval. Specially, a dPEKS scheme can be defined as follows.

Definition 1. *A dPEKS scheme consists of the following four PPT (probability polynomial-time) algorithms, (***Setup***,* **KeyGen***,* **dPEKS***,* **dTrapdoor***,* **dTest***).*

Setup: Let n be a security parameter. This algorithm takes n as input, then it outputs a set public parameter $\mathcal{PP}$.

KeyGen: Taking the public parameter $\mathcal{PP}$ as input, this algorithm creates the receiver's a public/private key pair (P_r, K_r) and the server's a public/private key pair (P_s, K_s).

dPEKS: Taking the public parameter $\mathcal{PP}$, the receiver's public key P_r, the server's public key P_s and a keyword w as input, this algorithm returns a dPEKS ciphertext C_w corresponding to w.

Trapdoor: Taking $\mathcal{PP}$, the receiver's public/private key (P_r, K_r), the server's public key P_s and a keyword w' as input, this algorithm generates a trapdoor T_w of w.

dTest: Taking a dPEKS ciphertext C_w of keyword w, $\mathcal{PP}$, a trapdoor $T_{w'}$ and the server's private key K_s as input, this algorithm returns 'yes' if $w' = w$, and otherwise outputs 'no'.

2.1.2 Security Model

Security of dPEKS ciphertext

As described in [15], in dPEKS, the security for a dPEKS ciphertext requires that a dPEKS ciphertext satisfies indistinguishability against a chosen plaintext attack (C-IND-CPA in short). Specially, the C-IND-CPA guarantees that (1) a server cannot distinguish between the dPEKS ciphertexts of two challenge keywords w_0 and w_1 its choice if he has not obtained their trapdoor. (2) an outside adversary (including a receiver) who can generate the trapdoors of any keyword (excluding challenge keywords) cannot distinguish between the dPEKS ciphertext of w_0 and w_1 its choice if he has not obtained the server's private key. Formalized, the C-IND-CPA can be defined with the following two games.

Game1. Here, $\mathcal{G}$ is a challenger and $\mathcal{F}_1$ is a malicious server.

Setup: $\mathcal{F}_1$ generates (P_s, K_s) as his public/private key pair. $\mathcal{G}$ generates (P_r, K_r) as receiver's public/private key pair. The tuples (P_s, K_s, P_r) are given to $\mathcal{F}_1$, and the tuples (P_r, K_r, P_s) are given to $\mathcal{G}$.

Phase 1 Trapdoor queries: $\mathcal{F}_1$ queries many keywords $w \in \{0,1\}^*$ to obtain trapdoors T_w from $\mathcal{G}$. $\mathcal{G}$ adaptively responses $\mathcal{F}_1$ with T_w as trapdoor generation oracle.

Challenge: $\mathcal{F}_1$ chooses the keywords pair (w_0, w_1) as a challenge. Here, the restriction is that w_0 and w_1 have not been queried to obtain the trapdoors T_{w_0} and T_{w_1}. Receiving w_0 and w_1, $\mathcal{G}$ chooses an random $b \in \{0,1\}$ and generates the ciphertext C_{w_b} of w_b, and returns it to $\mathcal{F}_1$.

Phase 2 Trapdoor queries: In this phase, $\mathcal{F}_1$ can still queries w to obtain its trapdoor as phase 1. If the $w \neq w_0, w_1$, $\mathcal{G}$ adaptively responses $\mathcal{F}_1$ with T_w as phase 1, otherwise stop.

Outputs: $\mathcal{F}_1$ outputs $c' \in \{0,1\}$. If $c' = c$, then $\mathcal{F}_1$ wins Game1.
Let $\mathbf{adv}_{\mathcal{F}_1}^{C-ind-cpa} = |Pr(c' = c) - \frac{1}{2}|$ denote the advantage probability that $\mathcal{F}_1$ wins the game1.

Game2. Here, $\mathcal{G}$ is a challenger and $\mathcal{F}_2$ an outside adversary (including receiver).

Setup: $\mathcal{F}_2$ is given P_r and K_r as receiver's public and private key, respectively. $\mathcal{G}$ (as server) generates (P_s, K_s) as his public/private key pair. The tuples (P_r, K_r, P_s) are given to $\mathcal{F}_2$, the tuples (P_s, K_s, P_r) are given to $\mathcal{G}$. Here, $\mathcal{F}_2$ can generate the trapdoor of any keyword since he holds K_r.

Challenge: $\mathcal{F}_2$ chooses the keywords pair (w_0, w_1) as the challenges. Here, the restrictions is that $\mathcal{F}_2$ did not previously ask the dTest oracle for the trapdoors of w_0 and w_1. Receiving w_0 and w_1, $\mathcal{G}$ chooses $c \in \{0,1\}$ and generates the ciphertext C_{w_c} of w_c, and returns it to $\mathcal{F}_2$.

Output: $\mathcal{F}_2$ outputs $c' \in \{0,1\}$. If $c' = c$, then $\mathcal{F}_2$ wins Game2.
Let $\mathbf{adv}_{\mathcal{F}_2}^{C-ind-cpa} = |Pr(c' = c) - \frac{1}{2}|$ denotes the advantage probability that $\mathcal{F}_2$ wins the Game2.

Definition 2. *For the polynomial-time $\mathcal{F}_1$ and $\mathcal{F}_2$, a dPEKS scheme is said to be C-IND-CPA secure if* $\mathbf{adv}_{\mathcal{F}_{1,2}}^{C-ind-cpa} = |Pr(c' = c) - \frac{1}{2}|$ *is negligible.*

Remark. In the Game2, the adversary is considered to be an receiver who can generate the trapdoor of keywords. If the outside adversary is not an receiver, we only need to the Game1 to define the C-IND-CPA. In fact, receiver's ability to discriminate between the dPEKS ciphertexts of keywords won't arise harmful effects, since the dPEKS ciphertexts should be send to receiver. Based on this reason, the adversaries are considered to be server and outside attacker (excluding receicer) when we prove the C-IND-CPA of BdPEKS.

Security of Trapdoor
As stated [15], in a dPEKS scheme, if the adversary (excluding the receiver and the server) cannot distinguish between the trapdoors of w_0 and w_1, it is said that a dPEKS scheme satisfies trapdoor indistinguishability against an adaptive chosen plaintext attack (T-IND-CPA). The dPEKS scheme can stand against outside keyword-guessing attacks successfully if it is T-IND-CPAT secure. The trapdoor indistinguishability can be defined with the following Game3.

Game3. Here, $\mathcal{G}$ is a challenger and $\mathcal{F}_3$ is an outside adversary.

Setup: Running **Setup** and **KeyGen**, the public parameter $\mathcal{PP}$, the receiver's key pair (P_r, K_r) and the server's key pair (P_s, K_s) are generated. $\mathcal{PP}$, P_r and P_s are given to $\mathcal{F}_3$ while K_s and K_r are kept secret from $\mathcal{F}_3$.

Phase 1 Trapdoor queries: $\mathcal{F}_3$ queries many keywords $w \in \{0,1\}^*$ to obtain trapdoors T_w from $\mathcal{G}$. $\mathcal{G}$ adaptively responses $\mathcal{F}_3$ with trapdoor T_w of w.

Challenge: $\mathcal{F}_3$ chooses (w_0, w_1) as challenge keywords and send them to $\mathcal{G}$. Here, the restriction is that w_0 and w_1 have not been queried to obtain the trapdoors T_{w_0} and T_{w_1}, and that $\mathcal{F}_3$ did not previously ask for T_{w_0} and T_{w_1} in phase 1. Receiving (w_0, w_1), $\mathcal{G}$ chooses an random $c \in \{0,1\}$ and generates its trapdoor T_{w_c}, and returns it to $\mathcal{F}_3$.

Phase 2 Trapdoor queries: In this phase, $\mathcal{F}_3$ can still query the trapdoor of w as phase 1, where $w \neq w_0, w_1$. $\mathcal{G}$ can adaptively response $\mathcal{F}_3$ with T_w as oracle.

Outputs: $\mathcal{F}_3$ outputs $b' \in \{0,1\}$. If $c' = c$, then $\mathcal{F}_3$ wins Game3
Let $\mathbf{adv}_{\mathcal{F}_3}^{T-ind-cpa} = |Pr(c' = c) - \frac{1}{2}|$ denote the advantage probability that $\mathcal{F}_3$ wins the Game3.

Definition 3. *For the polynomial-time $\mathcal{F}_3$, it is said to be T-IND-CPA if $\mathbf{adv}_{\mathcal{F}_3}^{T-ind-cpa} = |Pr(c' = c) - \frac{1}{2}|$ is negligible.*

2.1.3 Complexity Assumptions

Let G, G_T be multiplicative cyclic groups of prime order p.

The security of our system is based on the decisional augmented bilinear Diffie-Hellman exponent assumption (decisional ABDHE)[17]. First, we review the q-ABDHE problem, which is defined as follows. Let e be a bilinear map: $G \times G \rightarrow G_T$. Given a tuple in G^{2q+2}: $\mathcal{L}' = \left(\tilde{g}, \tilde{g}^{\gamma^{q+2}}, g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^q}, g^{\gamma^{q+2}}, \ldots, g^{\gamma^{2q}}\right)$ as input, required to output $e(g, \tilde{g})^{\gamma^{q+1}}$.

Following the q-ABDHE problem, the truncated version of q-ABDHE problem is defined as: Given a tuple $\mathcal{L} = \left(\tilde{g}, \tilde{g}_{q+2}, g, g_1, \ldots, g_q\right)$, required to output $e(g, \tilde{g})^{\gamma^{q+1}}$, where $\tilde{g}_i = \tilde{g}^{\gamma^i}$ and $g_i = g^{\gamma^i}$.

Clearly, the truncated q-ABDHE problem is hard if the q-ABDHE problem is hard. Corresponding to the truncated q-ABDHE problem, the decisional truncated q-ABDHE is introduced as follow.

An algorithm $\mathcal{G}$ outputs $b \in \{0, 1\}$ with the advantages ε in solving the truncated decision q-ABDHE if $|Pr[\mathcal{G}(\mathcal{L}, e(g_{q+1}, \tilde{g})] - Pr[\mathcal{G}(\mathcal{L}, E)]| \geq \varepsilon$, where the probability is over the random choice of γ in Z_p, the random choice of generators $g, \tilde{g}$ in G, the random choice of $E \in G_T$ the random bits consumed by $\mathcal{G}$.

Meanwhile, we also assume the discrete logarithm problem (DLP) assumption holds over G and G_T.

3 Our Construction

3.1 Basic dPEKS Scheme (BdPEKS)

In this section, we construct a basic searchable public-key encryption scheme with a designated tester (BdPEKS). With the G, G_T as specified above, let H_0: $\{0, 1\}^* \rightarrow Z_p^*$ be a hash function. Our scheme is built as follows.

- **Setup:** Taking two multiplicative cyclic groups G, G_T with prime order p, a bilinear map e, this algorithm generates public parameters $\mathcal{PP} = (p, G, G_T, g, \beta, e, H_0)$, where $g, \beta \in G$ and g is a generator of G.
- **KeyGen** (PP)**:** Taking a, b and $g \in \mathcal{PP}$ as input, this algorithm outputs the receiver's public key and private key $P_r = g^a$ and $K_r = a$, and the server's public key and private key $P_s = g^b$ and $K_s = b$.
- **dPEKS** $(P_r,\ w)$**:** Taking $\mathcal{PP}$, P_r and a keyword w (denote $H(w) = h$) as input, the sender chooses a random $u \in Z_p^*$ computes the dPEKS ciphertext as follows:
  $C = (C_1,\ C_2,\ C_3) = \left(e(\beta, g)^u,\ e(g, g)^u,\ P_r^u \cdot g^{-uh}\right)$
- **Trapdoor**$(K_r,\ P_s,\ w')$**:** Taking $K_r = a$, P_s ,a keyword w' (denote $H(w') = h'$) and a random $v \in Z_p^*$ as input, the receiver computes the trapdoor of w' as follows:
  $T = (T_1,\ T_2) = \left(P_s^v,\ T_2\ =\ g^v \cdot (\beta P_s^{-1})^{\frac{1}{a-h'}}\right)$.
- **dTest:** Given a dPEKS ciphertext C and a trapdoor T, the server performs searching operation by checking $C_1 = C_2^b \cdot e(C_3, T_2 T_1^{-(b^{-1})})$. If the equation holds, it returns 1; Otherwise returns 0;

3.1.1 Correctness of BdPEKS

The BdPEKS scheme is correct if the trapdoor $T = (T_1,\ T_2)$ is valid for w' and the dPEKS ciphertext $C = (C_1, C_2, C_3)$ is valid for w.

With $T = (T_1,\ T_2) = \left(P_s^v,\ g^v \cdot (\beta P_s^{-1})^{\frac{1}{a-h'}}\right)$ and $C = (C_1, C_2, C_3) = \left(e(\beta, g)^u, e(g, g)^u, P_r^u \cdot g^{-uh}\right)$, the correctness of the dTest algorithm is verified as follows: $T_2 \cdot T_1^{-b^{-1}} = (\beta P_s^{-1})^{\frac{1}{a-h'}}$, $e(C_3, T_2 T_1^{-b^{-1}}) = e(g^{u(a-h)}, \beta^{\frac{1}{a-h'}}) e(g^{u(a-h)}, g^{\frac{-b}{a-h'}}) = e(g^u, \beta) e(g^u, g^{-b})$.

Therefore, if $w = w'$, then the equation $C_1 = C_2^b \cdot e(C_3, T_2 T_1^{-b^{-1}})$ is holds.

3.1.2 Security of BdPEKS

Security of dPEKS Ciphertext

As stated in **Remark** of Sect. 2.1, the adversaries are considered to be a server or an outside attacker (excluding receiver).

Theorem 1. *For a server or an outside attacker (excluding receiver), if the truncated decision assumption holds over (G, G_T, e), the BdPEKS scheme is C-IND-CPA secure.*

Proof: We assume that the adversary $\mathcal{F}_1$ is the malicious server or an outside attacker, with an advantages ε breaking our scheme. We can construct an algorithm $\mathcal{G}$ which can solve the decisional truncated q-ABDHE problem on (G, G_T, e) with the advantage $(\varepsilon - 2/p)$.

Denote $g_i = g^{\alpha^i}$ and $\tilde{g}_i = \tilde{g}^{\alpha^i}$, $\mathcal{G}$ is given a random decision q-ABDHE challenge $(\tilde{g}, \tilde{g}_{q+2}, g, g_1, \cdots, g_q, E)$, where E is $e(g_{q+1}, \tilde{g})$ or a random element of G_T.

- **Setup:** $\mathcal{G}$ generates a random polynomial $d(x) \in Z_p[x]$ of degree q. Then $\mathcal{G}$ can compute $\beta = g^{d(\alpha)}$ since $(g, g^{\alpha}, \cdots, g^{\alpha^q})$. Then the public parameters are $\mathcal{PP} = (q, p, e, G, G_T, g, \beta, H)$, where $H{:}\{0,1\}^* \rightarrow Z_p^*$ is a hash function (not as oracle). Let $P_r = g_1$ be $\mathcal{G}$'s public key. Choose $b \in Z_p^*$ uniformly at random, then let $\mathcal{F}_1$'s public key and private key be $P_s = g^b$ and $K_s = b$.
- **Phase 1 Trapdoor queries:** The $\mathcal{F}_1$ makes queries of keywords $w \in \{0,1\}^*$ to obtain trapdoors T_w from $\mathcal{G}$. If $\mathcal{F}_1$ query the trapdoor of w ($h = H(w)$), $\mathcal{G}$ responds as follows. $\mathcal{G}$ computes the $(q-1)$-degree polynomial $D_T(x) = (d(x) - d(h))/(x - h)$. Taking two random r', $r'' \in Z_p^*$, he computes $T_1 = P_s^{r'}$ and $T_2 = g^{r''} g^{D_T(\alpha)}$. As an result, he sets $T = (T_1, T_2)$. Clearly, there is a unknown random r such that $r' = \frac{r}{\alpha - h}$ and $r'' = r' + \frac{d(h)-b}{\alpha - h}$. Thus $T_1 = P_s^{\frac{r}{\alpha - h}}$ and $T_2 = g^{r''} g^{D_T(\alpha)} = g^{\frac{r}{\alpha - h}} (\beta g^{-b})^{\frac{1}{\alpha - h}}$ appears to $\mathcal{F}_1$ be correctly distributed.
- **Challenge:** $\mathcal{F}_1$ chooses the keywords pair (w_0, w_1) as the challenge and send to $\mathcal{G}$. Denote $h_c = H(w_c)$ ($c \in \{0,1\}$). Here, the restriction is that w_0 and w_1 have not been queried to obtain the trapdoors T_{w_0} and T_{w_1}.

  Taking the polynomial $D_T(x)$, $d(x)$ and $d'(x) = x^{q+2}$, $\mathcal{G}$ computes $D'(x) = (d'(x) - d'(h_c))/(x - h_c)$, where the form of $D'(x)$ is $D'(x) = x^{q+1} + D(x)$. Then $\mathcal{G}$ picks $c \in \{0,1\}$ and computes the ciphertext as follows.
  Let $C = (C_1, C_2, C_3)$, then $C_2 = E \cdot e(\tilde{g}, g^{D'(\alpha)})$, $C_3 = \tilde{g}^{(d'(\alpha) - d'(h_c))}$
  $C_1 = e(C_3, g^{D_T(\alpha)}) \cdot C_2^{d(h_c)}$
  Let $u = (\log_g \tilde{g}) D'(\alpha)$, if $E = e(g_{q+1}, \tilde{g})$ then
  $C_1 = e(\beta, g)^u$, $C_2 = (g, g)^u$, $C_3 = g^{u(\alpha - h_c)}$. Since g, $\tilde{g}$ are uniformly random, the $u = (\log_g \tilde{g}) D'(\alpha)$ is uniformly random. As an result, the $C = (C_1, C_2, C_3)$ is a valid dPEKS ciphertext.

- **Phase 2 Trapdoor queries:** $\mathcal{F}_1$ makes trapdoor queries, for any keyword $w \neq w_0, w_1$, $\mathcal{G}$ responds as in Phase 1.
- **Guess:** Finally, $\mathcal{F}_1$ outputs it's result c'. If $c' = c$, $\mathcal{G}$ outputs 1(indicating $E = e(g_{q+1}, \tilde{g})$), otherwise outputs 0.

Clearly, if $E = e(g_{q+1}, \tilde{g})$, $\mathcal{F}_1$ can guess c correctly with the probability $1/2 + \varepsilon$. When E is uniformly random and independent element of G_T, the probability that $\mathcal{F}_1$ guesses c correctly is $2/p$. Meanwhile, the probability that $\mathcal{G}$ solves the truncated decision q-ABDHE correctly without $\mathcal{F}_1$'s help is $1/2$. As an result, $\mathcal{G}$ solves the truncated decision q-ABDHE with $1/2 + \varepsilon - 2/p - 1/2 = \varepsilon - 2/p$. This completes the proof of C-IND-CPA secure.

Security of Trapdoor

Theorem 2. *Our BdPEKS scheme is T-IND-CAP secure.*

Proof: The adversary $\mathcal{F}_2$ is assumed to be a malicious outside attacker. We show that $\mathcal{F}_2$ can not distinguish Whether two trapdoors were created by the same keyword.

Firstly, in our scheme, the trapdoor is $T_1 = P_s^v$, $T_2 = g^v \cdot (\beta P_s^{-1})^{\frac{1}{a-h'}}$ ($h' = H(w')$) where v is an random element in Z_p^*. The trapdoor is updated every time due to the difference of v we selected.

Due to the $K_r = a$ is kept secret from $\mathcal{F}_2$, the $\mathcal{F}_2$ can not known $(\beta P_s^{-1})^{\frac{1}{a-h'}}$. In fact, let $\beta = g^k$ ($k \in Z_p^*$ is some unknown value), then $T_2 = g^v \cdot (\beta P_s^{-1})^{\frac{1}{a-h'}} = g^{\frac{k-b}{a-h'}+v}$. With v is randomly selected from Z_p^*, T_2 is an random element in G. Thus T_2 is independent of keyword w' from $\mathcal{F}_2$'s view. As an result, our scheme satisfies the trapdoor indistinguishability.

3.2 Our Enhanced dPEKS Scheme (EdPEKS)

Base on the BdPEKS scheme, we construct an enhanced dPEKS scheme (EdPEKS). Our EdPEKS scheme has stronger security. Especially, the EdPEKS scheme can resist inside keyword guessing attacks from the untrusted server if the sender's identities are kept secret from cloud server. The EdPEKS scheme is constructed as follows.

Let G, G_T be multiplicative cyclic groups of prime order p. Let H_0: $\{0,1\}^* \to Z_p^*$ and H_1: $\{0,1\}^* \to G$ be two hash function.

- **Setup:** Take two multiplicative cyclic groups G, G_T with prime order p, a bilinear map e, this algorithm generates public parameters $\mathcal{PP} = (p, G, G_T, g, e, H_0, H_1)$, where $g \in G$ is a generator of G. Additionally, let $S_{id} \in \{0,1\}^*$ be the sender's identity.
- **KeyGen** ($\mathcal{PP}$)**:** Taking a, b and $g \in \mathcal{PP}$ as input, this algorithm outputs the receiver's public key and private key $P_r = g^a$ and $K_r = a$, and the server's public key and private key $P_s = g^b$ and $K_s = b$.
- **dPEKS** (P_r, P_s, S_{id}, w)**:** Taking $\mathcal{PP}$, P_r, P_s, S_{id} and a keyword w as input, the sender chooses a random $u \in Z_p^*$ computes the dPEKS ciphertext as follows:

$C=(C_1, C_2, C_3)=\big(e(H_{id}, P_s)^u, e(P_s, P_s)^u, P_r^u \cdot g^{-uh}\big)$
where $H_{id} = H_1(S_{id})$ and $h = H_0(w)$.

- **Trapdoor** (K_r, P_s, S'_{id}, w')**:** Taking $K_r = a$, P_s, S'_{id}, a keyword w' and a random $v \in Z_p^*$ as input, the receiver computes the trapdoor of w' as follows: $T = (T_1, T_2) = (P_s^v, g^v(H'_{id}P_s^{-1})^{\frac{1}{a-h'}})$, where $H'_{id} = H_1(S'_{id})$ and $h' = H_0(w')$
- **dTest:** Taking the dPEKS ciphertext C and trapdoor T, the server performs searching operation by checking $C_1 = C_2 \cdot e(C_3, T_2^b T_1^{-1})$. If this equation holds, it returns 1; Otherwise returns 0;

3.2.1 Correctness

The correctness of the EdPEKS scheme can be verified by the following equation.

With the trapdoor $T_1 = P_s^v$, $T_2 = g^v(H'_{id}P_s^{-1})^{\frac{1}{a-h'}}$ and the dPEKS ciphertext $C_1 = e(H_{id}, P_s)^u$, $C_2 = e(P_s, P_s)^u$, $C_3 = P_r^u \cdot g^{-uh}$, the server can compute:

$e(C_3, T_2^b T_1^{-1}) = e(P_r^u g^{-uh}, (H'_{id}P_s^{-1})^{\frac{b}{a-h'}}) = e\big(g^{u(a-h)}, H'^{\frac{b}{a-h'}}_{id}\big)e\big(g^{u(a-h)}, g^{\frac{-b^2}{a-h'}}\big)$

Clearly, if $H'_{id} = H_{id}$ and $w' = w$, the equation $C_1 = C_2 \cdot e(C_3, T_2^b T_1^{-1})$ holds.

3.2.2 Seacurity Analysis

In this section, we analysis the security of EdPEKS scheme.

Theorem 3. *Our EdPEKS scheme is the dPEKS ciphertext indistinguishable secure.*

Proof: Firstly, in EdPEKS scheme, the dPEKS ciphertext $C_3 = P_r^u \cdot g^{-uH_0(w)}$. Clearly, it is identical with the C_3 of Basic dPEKS scheme. If the adversary (the server or a outside attacker) can break the C-IND-CPA security of EdPEKS scheme, there exist an adversary can break the C-IND-CPA security of BdPEKS scheme.

Secondly, although the receiver may generate a trapdoor T, $P_s^{\frac{1}{a-H_0(w')}}$ and $g^{\frac{1}{a-H_0(w')}}$, he cannot perform a test since the server's private key K_s is kept secret from him. Therefore, the EdPEKS scheme is C-IND-CPA secure even if the adversary is receiver.

Theorem 4. *Our EdPEKS scheme is trapdoor indistinguishable secure.*

Proof: In EdPEKS scheme, the trapdoor $T = (P_s^v, g^v(H'_{id}P_s^{-1})^{\frac{1}{a-H_0(w')}})$. Clearly, the only difference between the trapdoor of EdPEKS scheme and the trapdoor of Basic scheme is that the β is replaced by H'_{id}. With the same analysis of theorem 3.2, it's easy to see that the EdPEKS scheme is also T-IND-CPA secure.

3.2.3 Inside KGA Analysis

As stated in [14], the inside KGA works as follows. Given a valid trapdoor, the server chooses an appropriate keyword from the keyword space and then uses it generate a dPEKS ciphertext. With his private key, the server can test whether the keyword matches the trapdoor. Since the keyword space is small, the *guessing-then-testing* procedure is efficient to find a correct keyword.

In our EdPEKS scheme, the dPEKS ciphertext is $\big(e(H_{id}, P_s)^u,\ e(P_s, P_s)^u,\ P_r^u \cdot g^{-uH(w)}\big)$. The EdPEKS scheme is secure against inside KGA due to the following reasons.

(1) It is easy to see that the server cannot obtain the sender's identity H_{id} from $e(H_{id}, P_s)^u$ even if he holds K_s and the trapdoor T.
(2) The server cannot obtain H_{id} from $T_2 = g^v \cdot (H'_{id} P_s^{-1})^{\frac{1}{a-H(w')}}$ since the receive's K_r is kept secret from him.
(3) The server cannot perform a test for a valid trapdoor if he has not the dPEKS ciphertext.

As a result, to launch KGA, he must guess the appropriate keyword and identity to computes a dPEKS ciphertext. With the space (including the identity space and the keyword space) becoming larger, the *guessing-then-testing* procedure is inefficient.

4 Performance Analysis

We analyze the performance of our schemes in terms of dPEKS ciphertext, the trapdoor and computation cost. This analysis includes a comparison between the other schemes.

Let P_t and E_t be the computational cost of a bilinear pairing operation and an exponentiation (or multi-exponentiation) over a bilinear group, respectively. Let l_G, l_{G_T}, l_p and l_H be the size of an element in G, G_T, Z_p^* and the hash value, respectively. Briefly, the size of dPEKS ciphertext and trapdoor denote ZC, ZT. In addition, the computation cost of trapdoor, ciphertext and test denote TrC, CiC and TeC.

In the Basic dPEKS scheme, by caching $e(\beta, g)$, $e(g, g)$, generating dPEKS ciphertext (C_1, C_2, C_3) does not need the pairing operation. Thus generating (C_1, C_2, C_3) only needs two exponentiations in G_T and one multi-exponentiation in G. Similar, in EdPEKS scheme, generating (C_1, C_2, C_3) need one pairing operation, one exponentiation in G_T and one multi-exponentiation in G. In Basic dPEKS scheme and EdPEKS scheme, generating the trapdoor need one exponentiation and one multi-exponentiation in G.

The Table 1 shows that only [16] and our EdPEKS scheme can resist inside KGA. Compared with others, our schemes are efficient.

Table 1. A comparison of various schemes

Schemes	ZC	ZT	TrC	TeC	CiC	Outside KGA	Inside KGA
[15]	$l_G + l_H$	$2l_G$	$2E_t$	$P_t + 2E_t$	$P_t + 2E_t$	Yes	No
[3]	$l_G + l_H$	l_G	E_t	$P_t + E_t$	$P_t + E_t$	No	No
[16]	$2l_G$	$2l_G + 2l_p$	$2E_t$	$2P_t + 2E_t$	$2E_t$	Yes	Yes
[14]	$3l_G + 2l_{G_T} + l_s$	$l_G + l_p$	E_t	$4P_t + 3E_t + t_v$	$3P_t + 6E_t + t_s$	Yes	No
[13]	$2l_G + l_{G_T}$	$2l_G$	E_t	$3P_t + E_t$	$P_t + 4E_t$	Yes	No
BdPEKS	$l_G + 2l_{G_T}$	$2l_G$	$2E_t$	$P_t + E_t$	$3E_t$	Yes	No
EdPEKS	$l_G + 2l_{G_T}$	$2l_G$	$2E_t$	$P_t + E_t$	$P_t + 3E_t$	Yes	Yes

5 Conclusion

In this paper, we proposed two dPEKS scheme, namely BdPEKS scheme and EdPEKS scheme. In BdPEKS scheme, we prove that the dPEKS ciphertext is C-IND-CAP secure without random oracle. Our BdPEKS scheme is secure against outside keyword-guessing attacks. The BdPEKS scheme is efficient because it only need multiplication and exponentiation to create a dPEKS ciphertext or a trapdoor. Under the original framework of [2], it is not possible to construct an dPEKS (PEKS) secure against inside KGA. To solve this problem, we proposed an enhanced dPEKS scheme (EdPEKS). With the sender's identity are kept secret from server, the EdPEKS scheme is secure resist inside KGA. In our EdPEKS, the trusted third party is removed. Both security analysis and compare results showed that the EdPEKS scheme is secure and efficient.

Acknowledgements. This study is supported by the National Natural Science Foundation of China (No. 61370203) and the Research Foundation of Education Bureau of Sichuan Province (12ZB348), China.

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of 2000 IEEE Symposium on Security and Privacy, S&P 2000, pp. 44–55. IEEE (2000)
2. Boneh, D., Boyen, X.: Efficient selective-ID Secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_14
3. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Gervasi, O., Murgante, B., Laganà, A., Taniar, D., Mun, Y., Gavrilova, M.L. (eds.) ICCSA 2008. LNCS, vol. 5072, pp. 1249–1259. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69839-5_96
4. Yau, W.-C., Heng, S.-H., Goi, B.-M.: Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Rong, C., Jaatun, M.G., Sandnes, F.E., Yang, L.T., Ma, J. (eds.) ATC 2008. LNCS, vol. 5060, pp. 100–105. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-69295-9_10
5. Xu, P., Jin, H., Wu, Q., Wang, W.: Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack. IEEE Trans. Comput. **62**(11), 2266–2277 (2013). IEEE
6. Fang, L., Susilo, W., Ge, C., Wang, J.: A secure channel free public key encryption with keyword search scheme without random oracle. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 248–258. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10433-6_16
7. Gu, C., Zhu, Y.: New efficient searchable encryption schemes from bilinear pairings. IJ Netw. Secur. **10**(1), 25–31 (2010)
8. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Comput. Commun. **32**(2), 394–396 (2009)

9. Liu, Q., Wang, G., Wu, J.: An efficient privacy preserving keyword search scheme in cloud computing. In: International Conference on Computational Science and Engineering, CSE 2009, pp. 715–720. IEEE (2009)
10. Rhee, H.S., Susilo, W., Kim, H.J.: Secure searchable public key encryption scheme against keyword guessing attacks. IEICE Electron. Expr. **6**(5), 237–243 (2009)
11. Yu, Y., Ni, J., Yang, H., Mu, Y., Susilo, W.: Efficient public key encryption with revocable keyword search. Secur. Commun. Netw. **7**(2), 466–472 (2014)
12. Hu, C., Liu, P.: A secure searchable public key encryption scheme with a designated tester against keyword guessing attacks and its extension. In: Lin, S., Huang, X. (eds.) CSEE 2011. CCIS, vol. 215, pp. 131–136. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23324-1_23
13. Zhao, Y., Chen, X., Ma, H., Tang, Q., Zhu, H.: A new trapdoor-indistinguishable public key encryption with keyword search. J. Wireless Mobile Netw. Ubiquit. Comput. Dependable Appl. **3**(1/2), 72–81 (2012)
14. Fang, L., Susilo, W., Ge, C., Wang, J.: Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf. Sci. **238**, 221–241 (2013)
15. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. **83**(5), 763–771 (2010)
16. Jiang, P., Mu, Y., Guo, F., Wang, X., Wen, Q.: Online/offline ciphertext retrieval on resource constrained devices. Comput. J. **59**(7), 955–969 (2015)
17. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006). https://doi.org/10.1007/11761679_27