Cluster Comput (2016) 19:1211–1217
DOI 10.1007/s10586-016-0584-7

Efficient public key encryption with revocable keyword search in cloud computing

Jianhong Zhang[1] · **Jian Mao**[2,3]

Received: 17 February 2016 / Revised: 11 May 2016 / Accepted: 13 June 2016 / Published online: 23 June 2016

Abstract Public key encryption with keyword search plays very important role in the outsourced data management. In most of public key encryption schemes with keyword search, the server can unlimitedly execute keyword search ability after obtaining a trapdoor information of a keyword. To restrict the ability of the server's unlimited search, we propose a novel public key encryption with revocable keyword search by combining hash chain and anonymous multi-receiver encryption scheme in this paper. The scheme can not only achieve security property of the indistinguishability of ciphertexts against an adaptive chosen keywords attack, but also resist off-line keyword guess attack. By comparison with Yu et al.'s scheme, our scheme is more efficient in terms of computational cost and communication overhead for the whole system.

Keywords Public key encryption · offline guessing attack · keyword search · security proof

1 Introduction

As a new type of computing paradigm, Cloud computing is becoming more and more concerns due to on-demand computing and higher cost-effective. To ensure the security of the data, a user need to encrypt the data before outsourcing in cloud computing. It will make the server doesn't know any information about the original data. However, it results in a serious problem: how can the user access the encrypted data by keyword search?

To solve the problem above, Song et al. proposed an efficient method to achieve search of the encrypted data in [1]. In this scheme, to guarantee data's confidentiality, data was encrypted by a symmetric encryption algorithm. In 2004, Boneh et al. studied the searching problem of the encrypted data by adopting public key cryptography technique, they put forward a novel notion: public key encryption with keyword search (PEKS) in [2]. And they showed that their PEKS scheme was secure against an active attacker who can obtain trapdoors information for any chosen keywords. However, the scheme exists a security issue: anyone who obtained several trapdoors information about unknown keywords can distinguish all captured encrypted data from the obtained trapdoor information. To solve the issue above, Baek et al. [3] introduced a designated entity to execute *Test* algorithm, and proposed a searchable public key encryption for designated tester (dPEKS)

In 2006, Byun et al. [4] firstly proposed off-line keyword guessing attack and pointed that some public key encryption schemes with keyword search were insecure against such attack. The main idea of this attack is based on an assumption which the used keywords in a PEKS are usually from a small keyword space. This enables an attacker to guess the validity of candidate keywords by adopting exhaustive search keyword space.

In most of the existing PEKS schemes [5–10], upon obtaining a trapdoor information of a keyword, the server can always search on the encrypted data. And it can utilize these trapdoor information to execute keywords guessing attack. In 2005,to solve the problem, Abdalla et al. [11]

✉ Jianhong Zhang
zjhncut@163.com

[1] College of Sciences, North China University of Technology, No.5 Jinyuanzhuang, Shijingshan District, Beijing 100144, People's Republic of China

[2] School of Electronic and Information Engineering, Beihang University, Beijing 100019, People's Republic of China

[3] The State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an 710071, People's Republic of China

proposed a public key encryption with temporary keyword search (PETKS) by using hierarchical identity based encryption. In the scheme, a trapdoor is only issued for any desired window of time rather than forever. The trapdoor's generation depends on a keywords w and time period t. Recently, for the server's unlimited keyword search power, Yu et al. proposed public key encryption with revocable keyword search (PERKS) in [12]. In this scheme, upon receiving trapdoor information of a keyword w in time period t, the server can test whether this keyword is present in the ciphertexts only under non-revoke statement. If the time period t is past, the server cannot test whether this keyword is present in the ciphertexts by using its owning trapdoor information in time period t.

In cloud computing environment, after a user outsourced a encrypted data file, for the user and the server, most frequent operations are *Test* and *revocation*. To construct a practical keyword search public key encryption in cloud computing, we propose a novel public key encryption with revocable keyword search in cloud computing in this paper. The advantages in our scheme are three-folds.

(1) Our scheme satisfies the indistinguishability of ciphertexts against an adaptive chosen keywords attack (IND-CKA) in the random oracle model.
(2) By comparison with Yu et al.'s scheme, our scheme has shorter ciphertext length, less computational cost in *Test* phase and *Revocation* phase.
(3) Our scheme can resist off-line keyword guessing attack.

2 Preliminaries

In this section, we review some fundamental background and mathematics problem which are used in this paper.

2.1 Bilinear pairings

Let G_1 be a cyclic additive group generated by P with the order prime q, and G_2 be a cyclic multiplicative group with the same order q. Let $e : G_1 \times G_1 \longrightarrow G_2$ be a pairing which satisfies the following conditions [2,11,12]:

- *Bilinear* For any $P, Q, R \in G_1$, we have $e(P+Q, R) = e(P, R)e(Q, R)$ and $e(P, R+Q) = e(P, R)e(P, Q)$. In particular, for any $a, b \in Z_q$,

$$e(aP, bP) = e(P, P)^{ab} = e(P, abP) = e(abP, P)$$

- *Non-degeneracy* There exists $P, Q \in G_1$, such that $e(P, Q) \neq 1$
- *Computability* There is an efficient algorithm to compute $e(P, Q)$ for $P, Q \in G_1$.

The bilinear map which satisfies the above properties is called an admissible bilinear map. They can be derived from the Weil pairing or the Tate pairing on an elliptic curve over a finite field.

2.2 Security assumption

Here, we give the corresponding mathematical hard problem on which our scheme is based.

2.2.1 The Co-DBDH problem

The Co-decisional Bilinear Diffie–Hellman problem in $\mathbb{G}_1$ is as follows: Given $(P, aP, bP, cP) \in \mathbb{G}_1^4$, $R \in \mathbb{G}_2$ for unknown $a, b, c \in Z_p$, to decide whether $R = e(P, P)^{abc}$.

2.3 Public key encryption with revocable keyword search (PERKS)

A PERKS scheme is a 5-tuple of polynomial-time algorithm (Setup, KenGen, PERKSEnc, TrapdoorGen, Test) where

- *Setup* The algorithm takes as input a security parameter l, and outputs public parameter $Params$ and the total number z of time periods.
- *KenGen* This algorithm takes as input the public parameter $Params$, and outputs a public-private pair (sk, pk) for the user.
- *PERKSEnc* The algorithm is run by the user. On inputting the public parameter $Params$, the public key pk, the current time period t and a keyword w, this algorithm outputs the ciphertext $C_{w,t}$ = PERKSEnc(Parmas,pk,w,t).
- *TrapdoorGen* This algorithm takes as input $Params$, sk, t and keyword w, and outputs the trapdoor information $T_{w,t} = TrapdoorGen(Params, sk, w, t)$ in the time period t.
- *Test* This is a deterministic algorithm. The algorithm takes as inputs $Params$, pk, the ciphertext $C_{w',t'}$ and trapdoor information $T_{w,t}$ in the time period t. And outputs 1,if $w = w'$ and $t' = t$ and 0, otherwise.

2.4 Security model

The security of a PERKS scheme denotes that it satisfies indistinguishability of PERKS against chosen keyword attack (IND-CKA). The security notion of IND-CKA guarantees that the server that has not obtained the trapdoors for given keywords cannot distinguish which PERKS ciphertext encrypts which keyword. Here, we define the security notion by the following games.

- *Setup* The challenger runs Setup algorithm and KeyGen algorithms to produce public parameter $Params$, the total number z of time period and public key pk, and sends them to the adversary $\mathcal{A}$. The challenger secretly keeps the private key sk.
- *Trapdoor queries 1* The adversary $\mathcal{A}$ makes a number of keyword queries to the challenger $\mathcal{C}$ as follows: upon receiving a keyword w and a time period t, $\mathcal{C}$ runs *TrapdoorGen* algorithm and returns trapdoor information $T_{w,t}$ to $\mathcal{A}$.
- *Challenge* $\mathcal{A}$ randomly chooses two challenged keywords w_0 and w_1 in the time period t and sends them to $\mathcal{C}$, where (w_0, t) and (w_1, t) cannot make the trapdoor queries 1. Upon receiving (w_0, t) and (w_1, t), $\mathcal{C}$ randomly a random $\beta \in \{0, 1\}$ to compute $C = PERKSEnc(Pramas, pk, w_\beta, t)$ and returns C to $\mathcal{A}$.
- *Trapdoor queries 2* $\mathcal{C}$ answers the trapdoor queries as in trapdoor queries 1. The restriction here is that (w_0, t) and (w_1, t) are not allowed to be queried.
- *Guess* Finally, $\mathcal{A}$ outputs its guess $\beta' \in \{0, 1\}$ and wins the game if $\beta = \beta'$.

The advantage of $\mathcal{A}$ is defined as $\mathrm{Adv}_{PERKS}^{IND-CKA}(\mathcal{A}) = \left|Pr[\beta' = \beta] - 1/2\right|$. The PERKS scheme is said to be (τ, ϵ)-IND-CKA secure if for any adversary $\mathcal{A}$, the guessing advantage $\mathrm{Adv}_{PERKS}^{IND-CKA}(A)$ is less than ε in polynomial time τ.

3 Keywords search public encryption with delegation

In this section, we will propose a novel public key encryption scheme with revocable keywords search. The main idea of our scheme is derived form Tseng et al.'s anonymous multi-receiver ID-based encryption scheme [13] and hash chain. we utilize receiver anonymity of Tseng et al.'s scheme to achieve indistinguishablity of chosen keywords attack. Our scheme consists of five algorithms: Setup,KeyGen, PERKSEnc, TrapdoorGen and Test. In the following, we give a detailed description.

Setup This algorithm takes a security parameter k as input, and outputs the following public parameters. $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic group with the same prime order q. P is a generator of group $\mathbb{G}_1$ and Q is a random element of group $\mathbb{G}_1$. $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ denotes a bilinear map. H_1 and H_2 are cryptographical hash functions which satisfies $H_1 : \{0, 1\}^* \to Z_q$ and $H_2 : \{0, 1\}^* \to \mathbb{G}_1$. H_3 is one-way hash function. The public parameters are published as follows.

$$Params = \{e, q, \mathbb{G}_1, \mathbb{G}_2, H_1, H_2, H_3, P, Q\}$$

The whole lifetime is divided into z parts in the system.

KenGen This algorithm takes as input public parameters $Params$, and outputs a public-private key pair $(pk, sk = s)$ where s is a random number of Z_q and $pk = sP$ is the corresponding public key. And it also chooses a random $r \in Z_q$ as the seed of hash chain.

PERKSEnc This algorithm takes as input public parameters $Params$, the public key pk and the selected keywords $\{w_1, w_2, \ldots, w_n\}$ and the current time period t, the ciphertext is produced as follows.

(1) Choose a random number $\gamma \in Z_q^*$ and compute $\beta = H_3^{z-t}(r)$, $C_1 = \beta\gamma P$ and $C_2 = \gamma pk$, where $H_3^t(r)$ denotes t times $H_3(r)$ operation, namely, $H_3^i(r) = H_3(H_3^{i-1}(r)), i = 1, \ldots, t$.
(2) For $i = 1$ to n, compute $v_i = H_2(e(H_1(w_i), C_2))$.
(3) Randomly choose a number $k \in Z_q$ and construct a polynomial $f(x)$ with degree n as below:

$$f_i(x) = \prod_{1 \le i \le n} (x - v_i) + k \mod q$$
$$= a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + x^n$$

where n denotes the number of the chosen keywords and $a_i \in Z_q, i = 1, \ldots, n$
(4) Then it encrypts r by the symmetrical key as $V = E_k(C_1)$.
(5) Finally, the resultant ciphertext is $C = (a_0, \ldots, a_{n-1}, C_1, V)$ and forward this ciphertext C the server.
(6) To revoke the search capability of the trapdoor T_{wt} in the tth time part, at the beginning of the $t + 1$-time, the user only needs to recompute

$$C_1' = \frac{H_3^{t+1}(r)}{H_3^t(r)} C_1$$

and

$$V' = E_k(C_1')$$

and stores the updated ciphertext $C' = (a_0, \ldots, a_{n-1}, C_1', V')$.

TrapdoorGen This algorithm takes as input public parameter $Params$, the private key s, a keyword w_j and the current time t, and outputs the following trapdoor information

(1) Compute $T_1 = \beta^{-1} s(H_1(w_j)$ and $T_2 = t$.
(2) Set $T_{wt} = (T_1, T_2)$ as the trapdoor information in the tth time.

Test In this algorithm, the server takes as input the public parameter $Params$, the trapdoor information T_{wt} and the ciphertext C, and computes

(1) Firstly compute $R = e(C_1, T_1)$
(2) Compute $v_j = H_2(R)$.
(3) Then construct the polynomial $f(x)$ by $(a_0, \ldots, a_{n-1})$ as

$$f(x) = a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$$

and computes $k = f(v_j)$
(4) Recover $C_1' = D_k(V)$ and check $C_1' \stackrel{?}{=} C_1$

If it holds, then it means that the keyword $w_j \in \{w_1, w_2, \ldots, w_n\}$ and returns the corresponding files which contain this keyword; otherwise, $w \notin \{w_1, w_2, \ldots, w_n\}$.

3.1 Correctness

With loss of generality, we suppose $w_j \in \{w_1, \ldots, w_n\}$. the correctness of the scheme is executed as follows.

$$\begin{aligned} e(C_1, T_1) &= e(\beta\gamma P, \beta^{-1} s H_2(w_j)) \\ &= e(\gamma P, s H_2(w_j)) \\ v_j &= H_2(e(\gamma P, s H_2(w_j))) \end{aligned}$$

$$\begin{aligned} f(v_j) &= a_0 + a_1 v_j + \cdots + c_{n-1} v_j^{n-1} + v_j^n \\ &= \prod_{1 \le i \le n} (v_j - v_i) + k \mod q \\ &= k \\ C_1' &= D_k(V) \\ C_1' &= ? = C_1 \end{aligned}$$

If $w_j \notin \{w_1, \ldots, w_n\}$, then the server can not correctly recover symmetrical key k. Thus, $C_1 \neq D_{k'}(V)$.

4 Security analysis

In the section, we will discuss the security of the proposed scheme and show that it is provable secure against the IND-CKA attack in the random oracle model.

Theorem 1 *If there exists an IND-CKA adversary $\mathcal{A}$ which can $(t, q_{H_1}, q_{H_2}, q_{H_3}, q_t, \epsilon)$-break our scheme, where q_{H_i} is the number of queries to random oracles $H_i (i = 1, 2, 3)$, q_t is the number of queries to trapdoor oracle, then there is another algorithm $\mathcal{C}$ who can solve an instance of the Co-DBDH problem by making use of $\mathcal{A}$ with probability*

$$\epsilon' \ge \left(\frac{1}{e(q_t + 1)}\right)^2 \epsilon$$

Proof Let $(P, aP, bP, cP, Z \in \mathbb{G}_2)$ be an instance of the Co-DBDH problem where $a, b, c \in Z_q$ are unknown, its goal is to compute $e(P, P)^{abc} \stackrel{?}{=} Z$.

Given an instance (P, aP, bP, cP, Z) of the Co-DBDH problem, $\mathcal{C}$ simulates a challenger and responses all the queries for $\mathcal{A}$.

- *Setup* $\mathcal{C}$ Sets $Q = aP$ and $pk = bP$ and randomly chooses a symmetric encryption algorithm (E_b, D_b) where b is a symmetric key. At the same time, it also choose a random number $r \in Z_q$ as the seed of hash function H_3. Finally, $\mathcal{C}$ sends public parameters $(\mathbb{G}_1, \mathbb{G}_2, e, P, pk, H_i, (i = 1, 2, 3), E, D)$ to the adversary $\mathcal{A}$. Hash function $H_i, (i = 1, 2)$ are random oracles controlled by $\mathcal{C}$. H_3 is an one-way hash function.
- H_1 *query* Upon receiving a query for H_1-oracle of keyword w_j, $\mathcal{C}$ first looks up H_1-list to check whether w_j was already defined in H_1-list. If it is , the corresponding value is returned to the adversary $\mathcal{A}$. Otherwise, it responses as follows:
  (1) $\mathcal{C}$ chooses a biased coin $c_i\{0, 1\}$ with the probability $Pr[c_i = 0] = 1 - \frac{1}{(q_t+1)}$.
  (2) Then, $\mathcal{C}$ picks a random $u_i \in Z_q$.
  If $c_i = 0, \mathcal{C}$ computes $y_i = u_i P$.
  If $c_i = 1, \mathcal{C}$ computes $y_i = u_i Q$.
  (3) Finally, $\mathcal{C}$ adds (w_i, u_i, y_i, c_i) to H_1-list and responds y_i to the adversary $\mathcal{A}$.
- H_2 *query* Upon receiving a query for H_2-oracle with $X_j \in \mathbb{G}_2$ for some $j \in [1, q_{h_2}]$, $\mathbb{C}$ first look up the H_2-list to check whether the input was already defined in the H_2-list. If it was, the corresponding defined value is returned to the adversary $\mathcal{A}$. Otherwise, $\mathcal{C}$ randomly picks a value $R_j \in Z_q$ and adds the tuple (X_j, R_j) into the H_2-list which is initially empty. Finally, $\mathcal{C}$ returns R_j to the adversary $\mathcal{A}$.
- **Trapdoor queries** 1: When the adversary $\mathcal{A}$ makes a trapdoor query with (w_i, t_i), $\mathcal{C}$ searches whether (w_i, u_i, y_i, c_i) is defined in the H_1-list. If it isn't, then $\mathcal{C}$ chooses a biased coin $c_i\{0, 1\}$. Then, $\mathcal{C}$ picks a random $u_i \in Z_q$.

  If $c_i = 0, \mathcal{C}$ computes $y_i = u_i P$.
  If $c_i = 1, \mathcal{C}$ computes $y_i = u_i Q$.

  and inserts (w_i, u_i, y_i, c_i) to H_1-list. Then $\mathcal{C}$ responses the trapdoor query by the following way:

If $c_i = 1$, $\mathcal{C}$ reports failure and aborts.
If $c_i = 0$, $\mathcal{C}$ sets $T_1 = (H_3^{t_i}(r))^{-1} u_i\, pk$ and $T_2 = t_i$. Thus, (T_1, T_2) is a valid trapdoor on keyword (w_i, t_i) in the t_i- time. Finally, $\mathcal{C}$ returns (T_1, T_2) to the adversary $\mathcal{A}$.

- **Challenge**: With loss of generality, $\mathcal{A}$ outputs two keywords w_0^* and w_1^* in the t^*-time as well as $n-1$ auxiliary keywords. The restriction condition are as follows:

  (1) (w_0^*, t^*) and (w_1^*, t^*) were not issued in trapdoor queries.
  (2) The return values of the used hash functions here had already been obtained from hash queries in the previous phase. Namely, $(w_0^*, u_0^*, v_0^*, c_0^*)$, $(w_1^*, u_1^*, v_1^*, c_1^*)$, $(w_2^*, u_2^*, y_2^*, c_2^*)$, $\cdots$, $(w_n^*, u_n^*, y_n^*, c_n^*)$ had existed in the H_1-list.

  If $c_0^* = 0$ or $c_1^* = 0$, then $\mathcal{C}$ reports failure and aborts. Otherwise, $\mathcal{C}$ executes the following steps:

  (1) $\mathcal{C}$ randomly picks $\zeta \in \{0, 1\}$ and sets $(w_\zeta^*, w_2^*, \ldots, w_n^*)$ as the challenged keywords.
  (2) Set $C_1 = H_3^{z-t^*}(r) cP$
  (3) For $i = 2$ to n, retrieve u_i from the tuple $(w_i^*, u_i^*, y_i^*, c_i^*)$ in the H_1-list and compute $v_i = H_2(e(U, u_i^* pk))$.
  (4) Compute $l = H_2(Z^{u_\zeta})$, where $(w_\zeta^*, u_\zeta^*, y_\zeta^*, c_\zeta^*)$ are defined in the H_1-list.
  (5) Compute and construct a polynomial $f(x)$ with degree n.

$$f(x) = (x-l)\sum_{2\le i\le n}(x - v_i) + k \mod q$$
$$= a_0 + a_1 x + \cdots + a_{n-1}x^{n-1} + x^n$$

  (6) Compute $V = E_k(C_1)$
  (7) Finally, set the ciphertext $C = (a_0, a_1, \ldots, a_{n-1}, C_1, V)$ and return C to the adversary $\mathcal{A}$.

- **Trapdoor queries 2:** $\mathcal{C}$ makes a number of trapdoor queries as in trapdoor queries 1. The only restriction condition here is that (w_0^*, t^*) and (w_1^*, t^*) are not allowed to issue the queries.
- **Guess:** Eventually, the adversary $\mathcal{A}$ outputs its guess $\zeta' \in \{0, 1\}$. If $\zeta = \zeta'$, then $\mathcal{C}$ outputs 1; otherwise, output 0.

According the above simulation, if $Z = e(P, P)^{abc}$, then we can know

$$\begin{aligned} Z^{u_\zeta} &= (e(P,P)^{abc})^{u_\zeta} \\ &= (e(aP, bP)^c)^{u_\zeta} \\ &= (e(aP, bP)^c)^{u_\zeta} \\ &= e(u_\zeta Q, bP)^c \\ &= e(H_2(w_\zeta^*), bP)^c \\ &= e(bH_2(w_\zeta^*), cP) \\ &= e(H_3(r)^{z-t^*} bH_2(w_\zeta^*),\ H_3(r)^{t^*-z} cP) \\ &= e(T_1, C_1) \end{aligned}$$

holds, It means that C is a valid ciphertext. Otherwise, Z is a random element of group $\mathbb{G}_2$, then C is an invalid ciphertext. If the adversary $\mathcal{A}$ can succeed in guessing the challenged ciphertext, then it can solve the Co-DBDH problem.

In the following, we evaluate the probability of solving the Co-DBDH problem. From the above simulation, we know if the following events appear, then $\mathcal{C}$ can correctly simulate without abortion.

- E_1 denotes $c_i = 0$ during the trapdoor queries.
- E_2 denotes $c_2^* = c_3^* = \cdots = c_n^* = 0$ during the challenged phase of the ciphertext.
- E_3 denotes $c_0^* = c_1^* = 1$ during the challenged phase of the ciphertext.

Supposed that the advantage of the adversary $\mathcal{A}$ is ϵ, then the probability of solving the Co-DBDH problem is

$$\begin{aligned} &Succ_{\mathbb{C}}^{Co-DBDH} \\ &= Pr[E_1 \wedge E_2 \wedge E_3]\epsilon \\ &= Pr[\bigwedge_{i=1}^{q_t} c_i = 0 \wedge (\bigwedge_{i=2}^{n} c_i^* = 0) \wedge (c_0^* = 1 \wedge c_1^* = 1)]\epsilon \\ &= Pr[\bigwedge_{i=1}^{q_t} c_i = 0] Pr[\bigwedge_{i=2}^{n} c_i^* = 0 | \bigwedge_{i=1}^{q_t} c_i = 0] \\ &Pr[(c_0^* = 1 \wedge c_1^* = 1) | \bigwedge_{i=2}^{n} c_i^* = 0, \bigwedge_{i=1}^{q_t} c_i = 0]\epsilon \end{aligned}$$

Because the maximum queries to trapdoor query is at most q_t, and the probability of $c_i = 0$ is $1 - \frac{1}{q_t+1}$, it means that $Pr\left[\bigwedge_{i=1}^{q_t} c_i = 0\right] = \left(1 - \frac{1}{q_t+1}\right)^{q_t}$. And the probability of event E_2' appearing is $\left(1 - \frac{1}{q_t+1}\right)^{n-1}$ under the condition of trapdoor queries without abortion. The probability of event E_3's appearing is $\left(\frac{1}{q_t+1}\right)^2$. Thus

$$\begin{aligned} &Succ_{\mathbb{C}}^{Co-DBDH} \\ &= \left(1 - \frac{1}{q_t+1}\right)^{q_t} \cdot \left(1 - \frac{1}{q_t+1}\right)^{n-1} \cdot \left(\frac{1}{q_t+1}\right)^2 \epsilon \\ &= \left(1 - \frac{1}{q_t+1}\right)^{q_t+n-1} \cdot \left(\frac{1}{q_t+1}\right)^2 \epsilon \end{aligned}$$

Table 1 Computation performance comparison of our scheme with Yu et al.'s scheme

Phase	Yu et al.'s scheme	Our scheme
KenGen	$PM_{\mathbb{G}_1}$	$PM_{\mathbb{G}_1}$
PERKSEnc	$(2n+2)PM_{\mathbb{G}_1} + nC_e + 1C_p$	$2PM_{\mathbb{G}_1} + nC_e + nC_p + C_{enc}$
TrapdoorGen	$PM_{\mathbb{G}_1}$	$PM_{\mathbb{G}_1}$
Revocation	$2nPM_{\mathbb{G}_1}$	$1PM_{\mathbb{G}_1}$
Test	$(2n-2)PM_{\mathbb{G}_1} + Mul_{\mathbb{G}_2} + 2C_e + 2C_p$	$1PM_{\mathbb{G}_1} + 1C_{enc}$

$$> \left(1 - \frac{1}{q_t+1}\right)^{2(q_t+1)} \cdot \left(\frac{1}{q_t+1}\right)^2 \epsilon$$
$$= \left(\frac{1}{e(q_t+1)}\right)^2 \epsilon$$

□

Theorem 2 *Our proposed scheme can resist keywords guessing attack.*

Proof In keywords guessing attack, we can divide the adversary into two types. One is the outsider attack, the other is insider attack, namely, the malicious server.

For the malicious server, it is more powerful than an outsider attacker since possess the encrypted data and a trapdoor information of a keyword. For the search of a keyword w, it can compute $R = e(H_1(w), pk)^\gamma$ and $v = H_2(R)$, and take v into polynomial $f(x)$ to compute $k = f(v)$.

For a guessed keyword w', if w' is a valid encrypted keyword, then it should satisfy $k = f(R')$ where $R' = e(H_1(w'), pk)^\gamma$. However, to compute R' by w' and pk, we must know γ. According to PERKS, we know that the related values to γ are C_1 and $R = e(H_1(w), pk)^\gamma$. For the malicious server, To solve r from C_1 or $R = e(H_1(w), pk)^\gamma$ is equal to the difficulty of solving discrete logarithm problem. Thus, our scheme can resist keywords guessing attack. □

5 Performance analysis

In this section, we discuss our scheme's performance in terms of computation and communication by comparing our scheme with Yu et al.'s scheme [12].

To assess computation performance, we define the following notation. Let $PM_{\mathbb{G}_1}$ denote the scalar point multiplication in group $\mathbb{G}_1$, $Mul_{\mathbb{G}_2}$ be multiplication in group $\mathbb{G}_2$, C_e be exponentiation operator in Z_q, C_p be pairing operator, and C_{enc} be symmetric encryption or decryption. Performance comparison of our schemes with Yu et al.'s scheme are summarized in Table 1.

To assess communication performance, we consider ciphertext message length, and define the following notation. Let $|P|$ denote the length of one elliptic curve point, $|Z_q|$ denotes the length of an element of Z_q. According to [14], to achieve a standard security level of 2^{80}, q is a 170-bit long prime number and each element in group $\mathbb{G}_1$ is 170 bits. Meanwhile, we suppose that the ciphertext length is 170 bits in a symmetric encryption. Communication performance comparisons between our scheme and Yu et al.'s scheme are summarized in Table 2.

Table 2 Communication performance comparison of our scheme with Yu et al.'s scheme

Phase	Yu et al.'s scheme	Our scheme
Ciphertexts size	$(2n+2) * 170$ bits	$(n+2) * 170$ bits
Trapdoor size	340 bit	170 bits

From Table 1, we can see that Yu et al.'s scheme has more advantage than our scheme in terms of computational cost in PERKSEnc phase. However, our scheme outperforms Yu et al.'s scheme in terms of computational cost in the *Test* phase. In cloud computing, the user only executes one time encryption operation before outsourcing the data and sends the outsourced data to the cloud server. To retrieve the data and restrict the server's search, Test algorithm and revocation algorithm are frequent used algorithms. The efficiency of these two algorithms directly influences the quality of the whole system. However, for these two algorithms, our scheme has more advantages over Yu et al.'s scheme.

As far as communication length, we can see that our scheme outperforms Yu et al.'s scheme from Table 2. In our scheme, the ciphertext length is $(n+2) * 170$ bits, however, the ciphertext length is $(2n+2) * 170$ bits in Yu et al.'s scheme.

According the statement above, for the performance of the whole system, our scheme outperforms Yu et al.'s in terms of computational cost and communication overhead. It can efficiently achieve keywords search and restrict the search of the server.

6 Conclusion

In most of the existing keyword search public key encryption, the server can execute the unlimited keywords search and off-line keywords guessing attack after obtaining the trapdoor information of keywords. To overcome these issues, we

proposed an efficient public key encryption with a revocable keyword search by utilizing hash chain technique and anonymous multi-receiver encryption. It can efficiently restrict search ability of the server. And the scheme is proven to be secure against an adaptive chosen keywords attack in the random oracle model. By comparison with Yu et al.'s scheme, our scheme has better advantage.

Acknowledgments This work was supported by Beijing Municipal Natural Science Foundation (Nos. 4162020, 4132056) and The importation and development of High-Caliber Talents project of Beijing municipal Institutions (CIT&TCD201304004).

References

1. Song, D., Wagner, D., Perrig, A.: Practical techniques for searching on encrypted data, In: Processing of the IEEE Symposium on Research in Security and Privacy. Berkeley, California, USA (2000), 44–55 (2000)
2. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: EUROCRYPT 2004, LNCS, vol. 3027, Springer: Interlaken, Switzerland, May 2–6 (2004), pp. 506–522
3. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Computational Science and Its Applications-ICCSA 2008. Lecture Notes in Computer Science, vol. 5072, pp. 1249–1259 (2008)
4. Byun, J.W., Rhee, H.S., Park, H.A., Lee, D.H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Jonker, W., Petkovic, M. (eds.) Secure Data Management, LNCS, vol. 4165, pp. 75–83. Springer, Berlin (2006)
5. Fan, C.I., Huang, L.Y., Ho, P.H.: Anonymous multireceiver ID-based encryption. IEEE Trans. Comput. **59**(9), 1239–1249 (2012)
6. Zhang, B., Zhang, F.G.: An efficient public key encryption with conjunctive-subset keywords search. J. Netw. Comput. Appl. **34**(1), 262–267 (2011)
7. Hu, C., Liu, P.: An enhanced searchable public key encryption scheme with a designated tester and its extensions. J. Comput. **7**(3), 716C723 (2012)
8. Mittelbach, F., Goossens, M.: The LaTeX Companion, 2nd ed. Addison-Wesley, Boston (2004)
9. Tang, Q., Chen, L.: Public-key encryption with registered keyword search. In: Martinelli, F., Preneel, B. (eds.) EuroPKI, vol. 6391. Lecture Notes in Computer Science, pp. 163–178. Springer, Berlin (2009)
10. Wang, B.J., Chen, T.H., Jeng, F.G.: Security improvement against malicious servers attack for a dPEKS scheme. Int. J. Inf. Educ. Technol. **1**(4), 350–353 (2011)
11. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. In: CRYPTO'05 (2005) 350–391
12. Yu, Y., Ni, J., Yang, H., Mu, Y., Susilo, W.: Efficient public key encryption with revocable keyword search. Secur. Commun. Netw. doi:10.1002/sec.790
13. Tseng, Y.-M., Huang, Y.-H., Chang, H.-J. Privacy-preserving multireceiver ID-based encryption with provable security. Int. J. Commun. Systems. doi:10.1002/dac.2395
14. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Proc. Adv. CryptologyłCrypto, ser. LNCS, vol. 3152. New York: Springer, 2004, pp. 41–45

Jianhong Zhang received his Ph.D. degrees in Cryptography from Xidian University, Xi'an, Shannxi, in 2004 and his M.S. degree in Computer Software from Guizhou University, Guiyang, Guizhou, in 2001. He was engaging in postdoctoral research at Peking University from October 2005 to December 2007. He has been an Assistant Professor of College of Sciences, North China University of Technology, Beijing, China, since 2001. His research interests include computer networks, cryptography, electronic commerce security, computer software.

Jian Mao is an Assistant Professor at Beihang University. My research interests include trust management and its applications, Web application analysis and cloud storage verification. I received my Ph.D. degree in Engineering from Xidian University in 2004, and B.S. degrees in Computer Comunications from Xidian University in 1997.