The Public Verifiability of Public Key Encryption with Keyword Search

Binrui Zhu[1], Jiameng Sun[1], Jing Qin[1,2(✉)], and Jixin Ma[3]

[1] School of Mathematics, Shandong University, Jinan, Shandong, China
zhubinrui1509889@163.com, sunjiameng1991@163.com
[2] State Key Laboratory of Information Security,
Institute of Information Engineering,
Chinese Academy of Sciences, Beijing, China
qinjing@sdu.edu.cn
[3] School of Computing and Mathematical Sciences,
University of Greenwich, London, UK
j.ma@greenwich.ac.uk

Abstract. Cloud computing has been widely recognized as the next big thing in this era. Users outsourced data to cloud server and cloud server provided service economic savings and various convenience for users. Public key encryption with keyword search (PEKS) which provides a solution for a third party user to search on remote data encrypted by data owner. Since the server may be dishonest, it can perform search operation on encrypted data and only return partial results. Therefore, it is necessary to verify the correctness and completeness of the search result. Existing PEKS schemes only support data receiver's private verification, however, in practice, we usually need anyone can verify the server's search result. In this paper, we propose a PEKS with public verifiability scheme, which can achieve the security of ciphertext indistinguishability, trapdoor indistinguishability, keyword guessing attack and public verifiability. Comparing previous PEKS schemes, our scheme is public verifiability, while keeping the encrypted data security in cloud server and search operation privately over the encrypted data.

Keywords: Cloud computing · PEKS · Public verifiability
Indistinguishability

1 Introduction

With the advent of the cloud ear, more and more users would like to store their data to the cloud server. By moving data to the cloud server, it provides both economical saving and various convenience for users. Despite having these benefits, security is still considered as the major barriers for the user and enterprise. Cloud server may be honest but curious, in order to ensure the security, data is usually stored as encrypted form in the cloud. At the same time, it also brings a new question that how users can get object encrypted data without decrypting

J. Hu et al. (Eds.): MONAMI 2017, LNICST 235, pp. 299–312, 2018.
https://doi.org/10.1007/978-3-319-90775-8_24

of them. Searchable encryption is a primitive, which enables data users to search over the encrypted data. Both keywords privacy and data privacy are protected in this procedure. PEKS provides a solution for the third party user to search on remote encrypted data and cloud server can return ciphertext corresponding the user's keywords. Thus, PEKS is suitable for three party situation application in cloud environment.

Considering a scenario: Patients upload the encrypted Personal Health Record (PHR) to the cloud server. Chief physician can search the patient's PHR information by keywords and its private key. PEKS can solve this background problem. But there are still two practical problems not solved by this method. Firstly, since the cloud server may be dishonest, which perform search operation on encrypted data and only return fraction information about the result. Secondly, if the cloud server perform search operation honestly, however, the Chief physician does not recognize the server to perform the search operation correctly on encrypted data. Traditional PEKS schemes can not solve these two questions. So, the PEKS scheme should support verifiable property, allowing the Chief physician can verify the cloud server whether executed the search operation correctly. At the same time, it also has the second question that how cloud server can prove that it performs the search operation honestly when the Chief physician maliciously denies the cloud server's search result. How to design a PEKS scheme to guarantee the confidentiality of PHR data and allow the search results can achieve public verifiability is a challenging problem.

Our contributions: We propose a PEKS with public verifiability scheme in which a data owner can encrypt message and keywords by the user public key, such that only the receiver who has private key can search the keyword in cloud environment and anyone can verify the cloud server whether returned the right result, which not just the data receiver can verify the search result. The most important thing is that the search process and verification process does not leak any information about the query and encrypted data. To the best of our knowledge, although a large body of PEKS with verifiability schemes have been proposed, few works have been done on PEKS scheme with public verifiability. Our scheme can achieve public verifiability while keeping the security of keywords indistinguishability, trapdoor indistinguishability and keyword guessing attack.

2 Related Works

Song et al. [1] proposed the first searchable encryption in 2000. This scheme is that the data owner uploads the encrypted data to the cloud server and searches by himself. But it can only support single keyword search and search requires linearly scan each file document word by word. The most important thing is that it is not fully secure and only supports user-server-user model. After this paper, many searchable encryption schemes [2–4] focusing on this model based on symmetric encryption. But these schemes are still unsuitable for three party situation. Symmetric searchable encryption schemes only supports user-server-user model, which is unsuitable in the cloud environment. Boneh et al. [5] proposed the first public key encryption with keyword search (PEKS) in 2004. Their

scheme provides a solution for the third party user to search on remote data encrypted by data owner. However, Boneh's scheme requires a secure channel and can not achieve indistinguish of trapdoor. Following Boneh's work, Baek et al. [6] proposed the notion of PEKS scheme without secure channel. Park et al. [7] proposed a new security model which is named public key encryption with conjunctive field keyword search. Abdalla et al. [8] proposed a general transformation from identity based encryption (IBE) to PEKS and the definition consistency of searchable encryption.

In order to resist the cloud server's dishonest behavior and return the incorrect search result, Chai and Gong [9] first proposed the concept of verifiable symmetric searchable encryption (VSSE) and given a formal VSSE definition of the protocol, including data owner and cloud server two participants. The data owner uploads the encrypted data, the cloud server performs the search operation, and the data owner receives the data search result and the search result proof. But it can only support single keyword search. Therefore, for improving the function of VSSE, Wang et al. [10] proposed a new VSSE scheme to support fuzzy keyword search. Zheng et al. [11] combines attribute encryption, digital signature, bloom filter, attribute keyword search and proposed a verifiable attribute based keyword search (VABKS) protocol which has good performance in search efficiency, but there are huge computational overhead in the verification process and can not resist offline attack. In the same year, Liu et al. [12] proposed a scheme which compared to the previous scheme, the verification algorithm has been greatly improved in efficiency. However, the Liu's scheme lacks integrity of the keyword detection in the verification process. Generally, the verifier is data owner and the practicality of the VSSE is limited in cloud environment. Many VSSE schemes [13,14] focusing on this model based on symmetric encryption. All of the above schemes are VSSE schemes and can not support public verifiability.

Alderman et al. [15] made a extension to Parno's verifiable computation scheme [16] from searchable encryption. They proposed a extended functionality in verifiable searchable encryption based on ciphertext policy attribute based encryption. The scheme not only supports more fine-grained keyword expression, but also achieves the public verifiability of search results. Compared to the previous scheme, Alderman's scheme has greatly improved the security and functionality, but the efficiency of the scheme is relatively low. Since the verifier needs to perform verify operation for each file to determine whether to meet the condition of the search query. Moreover, data owner and data receiver need interact with each other, only a data receiver who has private keys from data owner can search and decrypt the ciphertext in cloud environment. Zhang et al. [17] proposed a new public verifiable searchable encryption scheme, although the scheme can not achieve fine-grained search query, but the verification process has been greatly improved. Since the scheme requires secure channel and can not resist offline guessing attack and trapdoor indistinguishable, so the security model and the frame structure of the publicly verifiable searchable encryption are not complete. Meanwhile, Zhang's paper achieves public verifiability by signature for each item of index and files containing keyword, which also brings lower efficiency.

3 Preliminaries

In this section, we introduce the formal definition of the bilinear map and complexity assumptions. Let G_1, and G_T are two cyclic multiplicative groups of prime order p. A bilinear map $e : G_1 \times G_1 \rightarrow G_T$. Which satisfies:

1. Bilinear: For any $x, y \in Z_p$, $g \in G_1$, $e(g^x, g^y) = e(g, g)^{xy}$.
2. Non-degenerate: exist $g_1, g_2 \in G_1$, $e(g_1, g_2) \neq 1$.
3. Efficiency: There exits an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in G_1$.

Assumption 1 (HDH). The Hash Diffile-Hellman assumption: given the four tuple $(g, g^x, g^y, H(g^z))$ and hash function H, $x, y, z \in Z_p$, $G_1 =< g >$. It seems that x, y, and z is given to the adversary. If the adversary have the x, y, and z, decide whether $z = xy(modp)$ is not hard.

Assumption 2 (DBDH). The Decisional Bilinear Diffile-Hellman assumption: given the five tuple (g, g^x, g^y, g^z, Z), $x, y, z \in Z_p$, $Z \in G_T$, $G_1 =< g >$. It seems that x, y, and z is given to the adversary. If the adversary have the x, y, and z, decide whether $Z = e(g, g)^{xyz}$ is not hard.

Assumption 3 (BDH). The Bilinear Diffile-Hellman assumption: given the four tuple (g, g^x, g^y, g^z), $x, y, z \in Z_p$, $G_1 =< g >$. It seems that x, y, and z is given to the adversary. If the adversary have the x, y, and z, decide whether compute the value $(g, g)^{xyz} \in G_T$ is not hard.

Assumption 4 (BDHI). The Bilinear Diffie-Hellman Inversion assumption: given the two tuple (g, g^x), $x \in Z_p$, $G_1 =< g >$. It seems that (g, g^x) is given to the adversary. If the adversary have the (g, g^x), decide whether compute the value $e(g, g)^{\frac{1}{x}} \in G_T$ is not hard.

Assumption 5 (KEA1-r). The Knowledge of Exponent Assumption: given the three tuple (N, g, g^s) and returning (M, N) with $N = M^s$ to adversary $\mathcal{A}$, there exists extractor A', which given the same input as $\mathcal{A}$ returns d such that $M = g^d$.

Assumption 6 (DL). The Discrete logarithm assumption: given the two tuple (g, R), $x \in Z_p$, $G_1 =< g >$, $R \in G_1$. It seems that (g, R) is given to the adversary. If the adversary have the (g, R), decide whether find the only integer x which satisfy $g^x = R \ mod \ p$ is not hard.

4 PEKS

4.1 PEKS Model

Now, we will discuss the PEKS model. We consider the public-key scenario in which there are a data owner, an untrusted server, a data receiver, anyone verifier. The data owner encrypts index and data with receiver's public key and cloud server public key, the receiver can send the trapdoor to the server with

the receiver's secret key, so that the cloud server can perform search operation on encrypted data and data owner successfully authorizes data receiver's search ability through a public channel. A general PEKS scheme include five algorithms: Setup algorithm, Keygen algorithm, PEKS algorithm, Trapdoor algorithm, Test algorithm.

- **Setup**$(1^k) \rightarrow gp$: This algorithm inputs a security parameter 1^k, and generates a global parameter gp.
- **KeyGen**$(gp) \rightarrow (pk_s, sk_s, pk_r, sk_r)$: This algorithm inputs a global parameter gp, and generates two pairs of public and secret keys (pk_r, sk_r), (pk_s, sk_s) for receiver and server respectively.
- **PEKS**$(gp, pk_r, pk_s, w) \rightarrow C_w$: The data owner inputs global parameter gp, the receiver public key pk_r, the server public key pk_s, and the keyword w, outputs a ciphertext C_w for w.
- **Trapdoor**$(gp, pk_s, sk_r, w) \rightarrow T_w$: The data receiver inputs keyword w, the server public key pk_s, the receiver secret key sk_r, the global parameter gp, and computes trapdoor T_w corresponding the keyword w.
- **Test**$(T_w, C_w, sk_s) \rightarrow C_w$ *or* $\perp$: The server inputs the ciphertext C_w, a trapdoor T_w and the server secret key sk_s. It outputs the ciphertext C_w if $w = w'$, and $\perp$ otherwise.

4.2 Security Model

Definition 1 (Chosen keyword attack)**.** A PEKS scheme satisfies ciphertext indistinguishability secure against chosen keyword attack if for any probability polynomial time adversary $\mathcal{A}$, there is a neglible function $\epsilon(\lambda)$ such that

$$Adv^{cka}_{\mathcal{A}_{i,\{i\in 1,2\}}} = |Pr[b = b'] - 1/2| \leq \epsilon(\lambda).$$

A PEEKS scheme, we can define by the ciphertext indistinguishability experiment as follows:
*Game*1

- **Setup:** $\mathcal{A}_1$ is assumed to be a malicious server. The public parameter gp, the server key pairs (pk_s, sk_s) and the receiver public key pk_r are given to the $\mathcal{A}_1$.
- **Phase 1-1:** $\mathcal{A}_1$ makes the queries of the trapdoor T_w, adaptively makes any keyword queries for $w \in \{0,1\}^*$.
- **Phase 1-2:** $\mathcal{A}_1$ gives challenger $\mathcal{B}$ two be challenged keywords, w_0 and w_1. $\mathcal{B}$ randomly picks bit b, and computes a ciphertext for w_b and returns it to $\mathcal{A}_1$.
- **Phase 1-3**: $\mathcal{A}_1$ continues making trapdoor queries of the form w and the attacker can not ask for the trapdoors w_0 and w_1.
- **Phase 1-4:** $\mathcal{A}_1$ outputs its guess b'.

In this attack experiment, the advantage of the adversary $\mathcal{A}_1$ is:

$$Adv^{cka}_{\mathcal{A}_1} = |Pr[b = b'] - 1/2|.$$

*Game*2

- **Setup:** $\mathcal{A}_2$ is assumed to be a outside attacker. The public parameter gp, the receiver key pair (pk_r, sk_r) and the server public key pk_s are given to the $\mathcal{A}_2$.
- **Phase 1-1:** $\mathcal{A}_2$ makes the queries of the trapdoor T_w, adaptively makes any keyword queries for $w \in \{0,1\}^*$.
- **Phase 1-2:** $\mathcal{A}_2$ gives challenger $\mathcal{B}$ two be challenged keywords, w_0 and w_1. $\mathcal{B}$ randomly picks bit b, and computes a ciphertext for w_b and returns it to $\mathcal{A}_2$.
- **Phase 1-3**: $\mathcal{A}_2$ continues making trapdoor queries of the form w and the attacker can not ask for trapdoors w_0 and w_1.
- **Phase 1-4:** $\mathcal{A}_2$ outputs its guess b'.

About PEKS scheme, we can define the trapdoor indistinguishability experiment as follows:
*Game*3

- **Setup:** $\mathcal{A}$ is assumed to be an polynomial time attack algorithm, running time is bounded by t, the global parameter gp, the server public key pk_s and the receiver public key pk_r are given to the $\mathcal{A}$, keeping sk_r, sk_s secret.
- **Phase 1-1:** $\mathcal{A}$ makes the queries of the trapdoor T_w, adaptively makes any keyword queries for $w \in \{0,1\}^*$.
- **Phase 1-2:** $\mathcal{A}$ gives challenger $\mathcal{B}$ two be challenged keywords w_0 and w_1. $\mathcal{B}$ randomly picks bit b, and computes a trapdoor T_{w_b} for w_b and returns it to $\mathcal{A}$.
- **Phase 1-3:** $\mathcal{A}$ continues making trapdoor queries of the form w and the attacker can not ask for trapdoors w_0 and w_1.
- **Phase 1-4:** $\mathcal{A}$ outputs its guess b'.

The advantage of the adversary $\mathcal{A}$ in this game is defined as:

$$Adv_{\mathcal{A}}^{Trapdoorindistinguishability} = |Pr[b = b'] - 1/2|.$$

Definition 2 (Trapdoor indistinguishability)**.** A PEKS scheme is trapdoor indistinguishability secure for any probability polynomial time adversary $\mathcal{A}$, there is a neglible function $\epsilon(\lambda)$ such that

$$Adv_{\mathcal{A}}^{Trapdoorindistinguishability} = |Pr[b = b'] - 1/2| \leq \epsilon(\lambda).$$

5 A PEKS with Public Verifiability Scheme

In this section, we will propose a general construction method and security reduction for PEKS with public verifiability through study of PEKS. Data owners encrypt the keyword set with the receiver public key pk_r and cloud server public key pk_s. Data receiver generates the trapdoor with the receiver secret key sk_r and cloud server public key pk_s, and server matches the ciphertext with trapdoor and returns partial ciphertext file corresponding the keyword, any verifier can check the search result from server. Figure 1 shows the entire system framework. A PEKS with public verifiability scheme consists algorithms as follows:

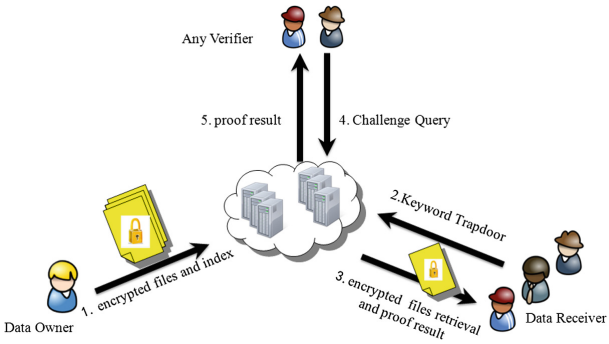

Fig. 1. PEKS with public verifiability

- **Setup**$(1^k) \rightarrow gp$: This algorithm inputs a security parameter 1^k, and generates a global parameter gp.
- **KeyGen**$(gp) \rightarrow (pk_s, sk_s, pk_r, sk_r)$: This algorithm inputs a global parameter gp, and generates two pairs of public and secret keys (pk_r, sk_r), (pk_s, sk_s) for receiver and server respectively.
- **TagGen**$(gp, pk_r, w) \rightarrow F_w$: The data owner inputs the receiver public key pk_r, the keyword w and computes the keyword tag F_w, releases F_w to be publicly known to everyone.
- **PEKS**$(gp, pk_r, pk_s, w) \rightarrow C_w$: The data owner inputs global parameter gp, the receiver public key pk_r, the server public key pk_s, and the keyword w, outputs a ciphertext C_w for w.
- **Trapdoor**$(gp, pk_s, sk_r, w) \rightarrow (T_w, ch_w)$: The data receiver inputs keyword w, the server public key pk_s, the receiver secret key sk_r, the global parameter gp, and computes trapdoor T_w and the challenge query ch_w corresponding the keyword w.
- **Test**$(T_w, C_w, sk_s, ch_w) \rightarrow C_w$ *or* $\perp$: The server inputs the ciphertext C_w, a trapdoor T_w, the challenge query ch_w and the server secret key sk_s. It outputs the ciphertext C_w and the challenge response R, if $w = w'$; and $\perp$ otherwise.
- **Private-Verify**$(T_w, R, gp) \rightarrow 1$ *or* 0: The data receiver inputs the global parameter gp, the challenge response R, and the return ciphertext C_w. It outputs the result 1, if the server performs search operation honestly, and 0 otherwise.
- **Public-Verify**$(T_w, F_w, gp) \rightarrow 1$ *or* 0: For public verifiably, we can divide into three steps:
  Challenge query$(gp) \rightarrow ch_w$: In order to verify the correctness file corresponding the keyword w, the public verifier inputs the global parameter gp. It outputs the challenge query ch_w.
  GenProof$(ch_w, gp, C_w) \rightarrow R$: The server inputs the return ciphertext C_w, the global parameter gp, the challenge query ch_w. It outputs the challenge response R.
  Verify$(F_w, R) \rightarrow 1$ *or* 0: The public verifier inputs the keyword tag F_w, the

challenge response R. It outputs the result 1, if the server performs search operation honestly, and 0 otherwise.

About PEKS with public verifiability scheme, we can define the public verifiability security experiment as follows:
*Game*4

- **Setup:** $\mathcal{A}$ is assumed to be an polynomial time attack algorithm and a malicious server, running time is bounded by t, the global parameter gp, the server key pair (pk_s, sk_s) and the receiver public key pk_r are given to the $\mathcal{A}$, keeping sk_r secret.
- **Phase 1-1:** $\mathcal{A}$ makes the queries of the trapdoor T_w and the challenge query ch_w, adaptively makes any keyword w. The challenger computes a keyword tag F_w and sends to the adversary.
- **Phase 1-2:** $\mathcal{A}$ outputs the ciphertext C'_w and the forge challenge response R'.
- **Phase 1-3:** The challenger outputs a bit $b = 1$, if the verification process can pass successfully, and 0 otherwise.

The advantage of the adversary $\mathcal{A}$ in this game is defined as:

$$Adv_{\mathcal{A}}^{Public\ Verifiability} = |Pr[b = 1] - 1/2|.$$

Definition 3 (Public verifiability)**.** A PEKS with public verifiability scheme is public verifiability secure for any probability polynomial time adversary $\mathcal{A}$, there is a neglible function $\epsilon(\lambda)$ such that

$$Adv_{\mathcal{A}}^{Public\ Verifiability} = |Pr[b = 1] - 1/2| \leq \epsilon(\lambda).$$

Since the verifier is not the data receiver himself, the scheme should ensure that any verifier can not get the private information about keywords and data file. We can define the security against any verifier by the simulation paradigm. Let f be an probabilistic polynomial time functionality and let $\prod$ be a two-party protocol for computing f. We denote the verifier V.

Definition 4 (Privacy against semi honest behavior [18])**.** f is a deterministic functionality, $\prod$ be a two-party protocol for computing f privately, if there exist probabilistic polynomial time algorithms, denoted S_1 and S_2.

$$\{S_1(x, f_1(x, y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{=} \{View_1^{\prod}(x, f_1(x, y))\}_{x,y\in\{0,1\}^*}.$$

$$\{S_2(x, f_V(x, y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{=} \{View_2^{\prod}(x, f_2(x, y))\}_{x,y\in\{0,1\}^*}.$$

By Definition 4, we can define the privacy against any verifier similarity, which is given in Definition 5.

Definition 5 (Privacy against any verifier)**.** A PEKS with public verifiability scheme is privacy against any verifier, for the keyword integrity checking protocol $\prod$, if there exists a probability polynomial time simulator S_v such that $\{S_v(x, f_V(x, y))\}_{x,y\in\{0,1\}^*} \stackrel{c}{=} \{View_V^{\prod}(x, f_V(x, y))\}_{x,y\in\{0,1\}^*}$.

6 A Concrete PEKS with Public Verifiability Scheme

6.1 Scheme Description

We will give a concrete PEKS with public verifiability scheme that is based on PEKS scheme. This scheme consists algorithms as follows:

- **Setup**$(1^k) \rightarrow gp$: This algorithm inputs a security parameter 1^k, and generates a global parameter $gp = (N, g, g_1, \eta, \sigma, H, H_1, H_2, G_1, G_T, f)$, letting $N = p_1 q_1$, $p_1 = 2p' + 1, q_1 = 2q' + 1$ are two large primes, p' and q' are primes, QR_N denotes the quadratic residues multiplicative cyclic group, which the generator is g_1, the order of g_1 is $p'q'$. G_1 and G_T are two cyclic multiplicative groups of prime order p, $g \in G_1$, several random elements $\eta, \sigma \in G_1$, $H : \{0,1\}^* \rightarrow G_1$, $H_1 : \{0,1\}^* \rightarrow G_1$, $H_2 : G_T \rightarrow \{0,1\}^k$, f is a pseudo-random function, $f : \{0,1\}^k \times \{0,1\}^{\log_2^l} \rightarrow \{0,1\}^d$, d is a security parameter.
- **KeyGen**$(gp) \rightarrow (pk_s, sk_s, pk_r, sk_r)$: This algorithm inputs a global parameter gp, chooses random numbersα and generates two pairs of public and secret key (pk_r, sk_r), (pk_s, sk_s) for receiver and server respectively, which $pk_r = (pk_{r_1}, pk_{r_2}) = (g^\beta, \eta^\beta)$, $sk_r = (\beta, p_1, q_1)$, $pk_s = (pk_{s_1}, pk_{s_2}) = (g^\alpha, \sigma^{\frac{1}{\alpha}})$, $sk_s = \alpha$.
- **TagGen**$(gp, j, w_i) \rightarrow (F_w)$: The data owner inputs global parameter gp, the keyword $w_i, i \in \{1, 2 \ldots n\}$, the file index number is j corresponding the keyword w_i and computes the keyword tag $F_{w_i} = \{F_{w_{ij}} = g_1^{H(w_{ij})} mod N\} = \{g_1^{H(w_i||j)} mod N\}, j \in \{1, 2 \ldots l\}$, releases F_w to be publicly known to everyone.
- **PEKS**$(gp, pk_r, pk_s, w) \rightarrow C_w$: The data owner inputs global parameter gp, the receiver public key pk_r, the server public key pk_s, the keyword w_i, and chooses random numbers $r \in Z_{p^*}$ outputs a ciphertext $C_{w_i} = [A, B, C] = [pk_{r_1}^r, H_2(e(H_1(w_i)^r, pk_{s_1})), F_{w_{ij}}]$ for w_i.
- **Trapdoor**$(gp, pk_s, sk_r, w) \rightarrow (T_w, ch_w)$: The data receiver inputs keyword w_i, the server public key pk_s, the receiver secret key sk_r, the global parameter gp, chooses random numbers $r' \in Z_p, t \in [1, 2^k - 1], s \in Z_N \setminus \{0\}$ and computes trapdoor $T_w = [T_1, T_2] = [g^{r'}, H_1(w)^{\frac{1}{\beta}} \cdot H(pk_{s_1}^{r'})]$ and the challenge query $ch_w =< t, g_s = g_1^s mod N >$ corresponding the keyword w.
- **Test**$(T_w, C_w, sk_s, ch_w) \rightarrow C_w$ *or* $\perp$: The server inputs the ciphertext C_w, a trapdoor T_w and the server secret key sk_s. If $B = H_2(e(A, (\frac{T_2}{H(T_1^{sk_s})})^{sk_s})$, it outputs the ciphertext $C_{w_i,j}$ and computes the coefficient $a_j = f_t(j)$, outputs the challenge response $R = (g_s)^{\sum_{j=1}^{l} a_j F_{w_{ij}}} mod N$, and $\perp$ otherwise.
- **Private-Verify**$(T_w, gp, R) \rightarrow 1$ *or* 0: The data receiver inputs the global parameter gp, the challenge response R and the return ciphertext C_w. It outputs the result 1, if the server honestly perform search operation which has $R' = g_1^{(\sum_{j=1}^{l} a_j F_{w_{ij}})s} mod\ N$ and $R = R'$, and 0 otherwise.
- **Public-Verify**$(T_w, pk_r, F_w, gp) \rightarrow 1$ *or* 0: For public verifiably, we can divide into three steps:

Public-challenge query$(gp) \rightarrow ch_w$: In order to verify the correctness file corresponding the keyword w, the public verifier inputs the global parameter gp, chooses random number $t \in [1, 2^k - 1], s \in Z_N \setminus \{0\}$. It outputs the challenge query $ch_w =< t, g_s = g_1^s mod\ N >$.
GenProof$(ch_w, gp, C_w) \rightarrow R$: The server inputs the return ciphertext C_w, the global parameter gp, the challenge query ch_w. It outputs the challenge response $R = (g_s)^{\sum_{j=1}^{l} a_j F_{w_{ij}}} modN$.
Verify$(F_w, R, gp) \rightarrow 1$ *or* 0: The public verifier inputs the global parameter gp, the keyword tag $F_{w_{ij}}$, the challenge response R. It outputs the result 1, if the server honestly perform search operation which has $R' = g_1^{(\sum_{j=1}^{l}(a_j F_{wij}))s} mod\ N$ and $R = R'$, and 0 otherwise.

Correctness: When assuming the ciphertext is valid for W' and the trapdoor T_W for W, we can verify query correctness as: $H_2(e(H_1(w_i)^r, pk_{s_1})) = H_2(e(A, (\frac{T_2}{H(T_1^{sk_s})})^{sk_s})$ test whether two values W and W' are equal. For the public verification correctness, we can verify correctness as:
$R' = g_1^{(\sum_{j=1}^{l} a_j F_{w_{ij}} mod\ N)s} mod\ N = R$, test whether two values R and R' are equal. Since the verifier can compute the coefficient $a_j = f_t(j)$ and the F_w to be publicly known. So, the verifier can verify the protocol correctness whether the server honestly perform protocol, include the server returns an empty set.

7 Security and Performance

7.1 Security Proof

In this section, we will prove the security of the protocol against the untrusted server and malicious receiver similar the paper [21–23]. Theorems guarantee that if the server return the complete search results, it can pass the private and public verifiability successfully.

Theorem 1: Suppose the BDH and BDHI problem is hard, the PEKS with public verifiability scheme is secure under selective keyword model attack.

Proof. About selective keyword attack, we need to resist two adversaries which are a malicious server adversary $\mathcal{A}_1$ and malicious outside adversary $\mathcal{A}_2$. We will proof the theorem similar paper [19], we will give the proof in the extended version, please see the full version of this paper.

Theorem 2: PEKS with public verifiability scheme is an secure scheme satisfies the trapdoor indistinguishability against a chosen keyword attack, under assumption that Hash Diffie-Hellman (HDH) is intractable.

Proof. Security proof similar paper [19], we will give the proof in the extended version, please see the full version of this paper.

Theorem 3: PEKS with public verifiability scheme is an secure scheme against the untrusted server, under the KEA1-r and the large integer factorization assumption.

Proof. Adversary $\mathcal{A}$ is an polynomial time attack algorithm can break the PEKS with public verifiability scheme with the advantage ε. We construct an algorithm $\mathcal{B}$ that solves the integer factorization problem with ε'. Algorithm $\mathcal{B}$ is given a larger integer $N = p_1 q_1$. It's goal is to output two large prime number p_1 and q_1. Algorithm $\mathcal{B}$ simulates the challenger and interacts with adversary $\mathcal{A}$. Since the space restrictions, the simulation will in the full version.

Adversary $\mathcal{A}$ forge an requested keyword response R to the algorithm $\mathcal{B}$. If the verify (F_w, R, pk_r, gp) can pass successfully, the adversary $\mathcal{A}$ forge R successfully. Let we analyze the integer factorization problem. Adversary $\mathcal{A}$ has the tuple (N, g, g^s) and outputs the response $R = (g_s)^{\sum_{j=1}^{l} a_j F_{wij}}$, $a_j = f_t(j)$ for $j \in \{1, 2 \ldots l\}$. Because $\mathcal{A}$ can naturally computes $P = g_1^{\sum_{j=1}^{l} a_j F_{wij}}$, so $\mathcal{A}$ has two tuple $(R = P^s, P)$. Since the knowledge of exponent assumption, we can construct an extractor $\mathcal{A}'$ and output d which satisfy $P = g^d mod N$, Algorithm $\mathcal{B}$ can obtain $d = \sum_{n=1}^{j} a_j H(w_i \parallel j) \mod p'q'$.

Since the space restrictions, the equation solution will in the full version.

By solving the above equation, algorithm $\mathcal{B}$ can get $H^*(w_{ij}) = H(w_{ij}) \mod p'q'$, for each $j \in \{1, 2 \ldots l\}$. If $H^*(w_{ij}) = H(w_{ij})$, algorithm $\mathcal{B}$ can extract all the keyword Hash function $H(w_{ij})$, otherwise, there exist j, $H^*(w_{ij}) \neq H(w_{ij})$, then algorithm $\mathcal{B}$ can compute the integer prime factorization about N. Because $H^*(w_{ij}) = H(w_{ij}) \mod p'q'$, algorithm $\mathcal{B}$ can obtain a multiple of $\phi(N)$, so $\mathcal{B}$ can compute the integer prime factorization about N from the lemma 1 in [20]. Because of the difficulty of the integer prime factorization, we can see that any keyword Hash function can be extracted. Overal, the proposed scheme guarantees the keyword integrity against an untrusted server.

Theorem 4: PEKS with public verifiability scheme is an secure scheme against the third party verifier, under the semi-honest model.

Proof. In this proof, we can prove the scheme security against the third party verifier by a two party protocol for computing in the semi-honest model. Security proof similar paper [21]. The verifier and the server are denoted by V and P respectively. The view of the third party verifier is denoted as $View_V^P$. Exists a probability polynomial time simulator S_v for the view of the verifier. Our goal is to prove that the output of the simulator S_v is computationally indistinguishable the $View_V^P$ of the verifier. The verifier has the tuple $(N, g, \{F_{w_i, i=1,2\ldots n}\})$ as the input and output a bit b as result which denotes the success or failure. The simulator's input and output is similar with the verifier. If the server is honest, the bit b is always 1. Since the space restrictions, the simulation will in the full version.

In conclusion, we have the formula $\{S_v(x, f_V(x, y))\} \stackrel{c}{=} \{View_V^{\prod}(x, f_V(x, y))\}$ is proved, $x, y \in \{0, 1\}^*$. So the verifier cannot get any information from the search result, except its input and output.

Theorem 5: PEKS with public verifiability scheme is an secure scheme which can achieve public verifiability, under assumption that DL is intractable.

Proof. To proof our PEKS with public verifiability scheme is an secure scheme which can achieve public verifiability, we can discuss that the adversary $\mathcal{A}$ forge an requested keyword response R and verify $(F_w, R, gp) = 1$. According to equation $R = R'$, we can compute the equation probability $\Pr[(g_s)^{\sum_{j=1}^{l} a_j F_{wij}} = g_1^{(\prod_{j=1}^{l}(a_j F'_{wij}))s}]$. Let's simplify it further, $\Pr[H(w'_i \parallel j) = H(w_i \parallel j)], j = 1, 2 \ldots l$. Since the H is a collision-resistant Hash function, according to the Hash function definition, the adversary can not find a keyword w'_i such that $w'_i \neq w_i$ and $H(w'_i \parallel j) = H(w_i \parallel j), j = 1, 2 \ldots l$. Besides the equation $\Pr[H(w'_i \parallel j) = H(w_i \parallel j)], j = 1, 2 \ldots l$, the adversary need extractor the $H(w'_i \parallel j)$ from $g^{H(w'_i \| j)}$ is also a hard problem in DL assumption.

About the security keyword guessing attack, we add public key and private key for the cloud server and the data owner can re-encrypt the keyword ciphertext and trapdoor in public channel. This guarantees only the cloud server can match the keyword ciphertext and trapdoor. Our scheme can effectively prevent the guessing attack by this approach.

7.2 Security and Performance Analysis

Basic operations are recorded as: Let E denote an exponentiation operation, P is the basic operation of hash operation, M denote a multiplication operation in the group, e denote a pairing operation, f denote a polynomial operation and k represent the maximum number of trapdoor. Tables 1 and 2 give us the comparison between our scheme and the previous public key encryption with keyword search scheme. We use Trap Ind, Ciph Ind, PV, KS, KR to denote Trapdoor indistinguishability, Ciphertext indistinguishability, Public verifiability, Keyword guessing attack, KeyGenServer, KeyGenReceiver.

Table 1. Security comparison

	Boneh et al.	Baek et al.	Yang et al.	Our.
Trap Ind	No	No	Yes	Yes
Ciph Ind	Yes	Yes	Yes	Yes
KG	No	No	Yes	Yes
PV	No	No	No	Yes

Table 2. Performance comparison

	Boneh et al.	Baek et al.	Yang et al.	Our.
KS	-	M	-	2E
KR	E	M	6E	2E
PEKS	2E + 2P + e	E + M + P + 2e	(2k + 6)E	2E + P + e
Trapdoor	E + P	P + M	4f	4E + 2P + M
Test	e + P	M + e	2E + 4f	lf + 3E + P + e

8 Conclusions

In order to overcome the disadvantages of traditional PEKS in public verifiability environment, this paper proposes a PEKS with public verifiability scheme by using bilinear technique. By the security analysis, we have proved that the scheme achieves keyword indistinguishability, trapdoor indistinguishability, keyword guessing attack, public verifiability. Comparing with existing schemes, our scheme supports the public verifiability in cloud environment. It is a suitable method for solving the practical problem which is described in the introduction. Through the aforementioned content, we can get that this proposed the public verifiability of public key encryption with keyword search scheme is a secure and wide applicable protocol, and has a certain practical value.

Acknowledgment. This work is supported by the National Nature Science Foundation of China under Grant No: 61272091 and No: 61772311.

References

1. Song, D.X., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, pp. 44–55. IEEE Computer Society (2000)
2. Goh, E.J.: Secure indexes. IACR Cryptol. ePrint Arch. **2003**, 216 (2003)
3. Reza, C., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definition and efficient constructions. In: 2006 Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 79–88. ACM (2006)
4. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005). https://doi.org/10.1007/11496137_30
5. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_30
6. 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
7. Park, D.J., Kim, K., Lee, P.J.: Public key encryption with conjunctive field keyword search. In: Lim, C.H., Yung, M. (eds.) WISA 2004. LNCS, vol. 3325, pp. 73–86. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31815-6_7
8. Abdalla, M., Bellare, M., Catalano, D., et al.: Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J. Cryptol. **21**(3), 350–391 (2008)
9. Chai, Q., Gong, G.: Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In: 2012 IEEE International Conference on Communications, pp. 917–922. IEEE (2012)

10. Wang, J., Ma, H., Tang, Q., et al.: Efficient verifiable fuzzy keyword search over encrypted data in cloud computing. Comput. Sci. Inf. Syst. **10**(2), 667–684 (2013)
11. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: INFOCOM, 2014 Proceedings IEEE, pp. 522–530. IEEE (2014)
12. Liu, P., Wang, J., Ma, H., et al.: Efficient verifiable public key encryption with keyword search based on KP-ABE. In: 2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications (BWCCA), pp. 584–589. IEEE (2014)
13. Wei, X., Zhang, H.: Verifiable multi-keyword fuzzy search over encrypted data in the cloud. In: 2016 International Conference on Advanced Materials and Information Technology Processing, pp. 271–277 (2016)
14. Nie, X., Liu, Q., Liu, X., Peng, T., Lin, Y.: Dynamic verifiable search over encrypted data in untrusted clouds. In: Carretero, J., Garcia-Blas, J., Ko, R.K.L., Mueller, P., Nakano, K. (eds.) ICA3PP 2016. LNCS, vol. 10048, pp. 557–571. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-49583-5_44
15. Alderman, J., Janson, C., Martin, K.M., Renwick, S.L.: Extended functionality in verifiable searchable encryption. In: Pasalic, E., Knudsen, L.R. (eds.) Balkan-CryptSec 2015. LNCS, vol. 9540, pp. 187–205. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-29172-7_12
16. Parno, B., Raykova, M., Vaikuntanathan, V.: How to delegate and verify in public: verifiable computation from attribute-based encryption. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 422–439. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_24
17. Zhang, R., Xue, R., Yu, T., et al.: PVSAE: a public verifiable searchable encryption service framework for outsourced encrypted data. In: 2016 IEEE International Conference on Web Services, pp. 428–435. IEEE (2016)
18. Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge (2004)
19. Rhee, H.S., Park, J.H., Susilo, W., et al.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. **83**(5), 763–771 (2010)
20. Miller, G.L.: Riemann's hypothesis and tests for primality. J. Comput. Syst. Sci. **13**(3), 300–317 (1976)
21. Hao, Z., Zhong, S., Yu, N.: A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Trans. Knowl. Data Eng. **23**(9), 1432–1437 (2011)
22. Alabdulatif, A., Kumarage, H., Khalil, I., et al.: Privacy-preserving anomaly detection in cloud with a lightweight homomorphic approach. J. Comput. Syst. Sci. **90**, 28–45 (2017)
23. Kumarage, H., Khalil, I., Alabdulatif, A., et al.: Secure data analytics for cloud-integrated internet of things applications. IEEE Cloud Comput. **3**(2), 46–56 (2016)