Guo *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2022) 11:14
https://doi.org/10.1186/s13677-022-00287-5

Journal of Cloud Computing: Advances, Systems and Applications

RESEARCH Open Access

An improved secure designated server public key searchable encryption scheme with multi-ciphertext indistinguishability

Junling Guo[1], Lidong Han[1,2*], Guang Yang[1], Xuejiao Liu[1,2] and Chengliang Tian[3]

Abstract

In the cloud, users prefer to store their sensitive data in encrypted form. Searching keywords over encrypted data without loss of data confidentiality is an important issue. In 2004, Boneh et al. proposed the first public-key searchable encryption scheme which allows users to search by the private key. However, most existing public-key searchable encryption schemes are vulnerable to keyword guessing attack and can not satisfy multi-ciphertext indistinguishability. In this paper, we construct a secure designated server public-key searchable encryption based on Diffie-Hellman problem. Our security analysis shows that our proposed scheme can resist against keyword guessing attack and provide multi-ciphertext indistinguishability for any adversity. Furthermore, the proposed scheme can achieve multi-trapdoor privacy for external attackers. Moreover, the simulation results between our scheme and previous schemes demonstrate our new scheme is suitable for practical application.

Keywords: Searchable encryption, Keyword guessing attack, Multi-ciphertext indistinguishability, Diffie-Hellman problem, Multi-trapdoor privacy

Introduction

With the rapid development of cloud computing, a growing number of users and companies prefer to store data on the cloud. In such case, they encrypt the data before uploading in order to ensure data privacy. However, it is extremely difficult to retrieve keyword over encrypted data using traditional search mechanism. Searchable encryption has become a promising solution to ensure the security and availability of data.

In 2004, Boneh et al. [1] proposed the concept of Public-key Encryption with Keyword Search (PEKS) and gave a concrete scheme. However, in 2006, Byun et al. [2] put forward an offline keyword guessing attack(KGA) against Boneh et al. 's scheme. Later, Baek et al. [3] presented a PEKS scheme without a secure channel in 2008. Then, Rhee et al. [4] introduced a new security concept of PEKS, trapdoor indistinguishability, They put forward a PEKS scheme under designated test server (dPEKS) which satisfies trapdoor indistinguishability.

Wang et al. [5] proposed that even if [4] satisfies the trapdoor indistinguishability, their dPEKS cannot resist inside KGA. Since keyword encryption algorithms are public in previous schemes, it will enable the internal attacker to generate the ciphertext of a candidate keyword by himself. That is, the malicious server can efficiently test whether the trapdoor is generated by the canditate keyword or not.

To resist keyword guessing attacks initiated by malicious servers, many researchers have proposed some variants of PEKS schemes. Tang et al. [6] introduced the concept of keyword registration, which requires the sender to register keywords with the receiver in advance and proposes registered keyword search public key encryption (PERKS). Chen et al. [7] put forward a solution using two servers that do not collide with each other, but it is too ideal. Later, Huang et al. [8] presented the concept of public-key

*Correspondence: ldhan@hznu.edu.cn
[1]School of Information Science and Technology, Hangzhou Normal University, Hangzhou, China
Full list of author information is available at the end of the article

authenticated encryption with keyword search (PAEKS) to resist the inside KGA. In their scheme, the data owner needs to use the secret key to authenticate the ciphertext of the keyword. The malicious cloud server will not generate keyword ciphertext for testing without the owner's private key. Therefore, KGA does not succeed against their scheme.

Qin et al. [9] in 2020 introduced the new security concept called multi-ciphertext indistinguishability (MCI). That is, from two or more ciphertexts, the adversary can determine whether they are generated by a same keyword. And they constructed a new PAEKS that can guarantee MCI security but does not provide multi-trapdoor privacy (MTP) security in which attacker is able to check two or more trapdoors contain a same keyword. In 2021, Pan and Li [10] put forward a new PAEKS scheme with MCI and MTP security. Later, Cheng and Meng [11] proved that Panr and Li's scheme does not satisfy MTP security.

Motivations and contributions

In searchable encryption, the security goal is that the ciphertexts and trapdoors leak no information about keywords. So far, there is rarely public-key searchable encryption schemes achieve both MCI and MTP, and security against KGA. In this paper, our goal is to construct an enhanced secure designated server public-key searchable encryptionscheme with MCI and MTP. The contributions of our paper are summarized as follows:

1 We give a security analysis of Li et al.'s scheme [12] and show that their scheme does not satisfy trapdoor indistinguishability.
2 We propose a secure scheme that satisfies the requirement of testing the designated server. That is to say, no one can test except the designated server. Moreover, we prove that our scheme satisfies MCI security, MTP security for external adversaries, and designated testability.
3 We analyze our scheme's implementation and communication cost by comparing it with previous other schemes. The result shows that our scheme has excellent advantages in keyword ciphertext and trapdoor algorithms, and the test algorithm is not inferior to other schemes. Moreover, our scheme provides stronger security for keyword privacy.

Related works

In 2004, Boneh et al. [1] first proposed the public key encryption scheme with keyword search, which started the research on public-key searchable encryption. Later, Abdalla et al. [13] presented a searchable encryption scheme based on identity. Byun et al. [2] put forward offline KGA against Abdalla et al.'s scheme. Baek et al. [3] suggested that a tester should be appointed to perform the test algorithm to hide the user's search pattern, to ensure that only those who have the tester's private key can conduct the test. Rhee et al. [4, 14] put forward a dPEKS model to reject outside KGA and constructed a general structure of dPEKS based on the designated tester. Fang et al. [15] presented a dPEKS scheme that is not based on a random prediction machine to resist outside KGA. Rhee et al. [16] construct an identity-based PEKS scheme with a designated tester. Emura et al. [17] presented a general structure of SCF-PEKS based on anonymous identity-based encryption(IBE) and one-time signature. After that, many schemes [18–20] have made efforts to resist offline guessing attacks, but these schemes cannot resist inside KGA.

To resist inside KGA, Xu et al. [21] proposed a PEKS scheme with fuzzy keywords, reducing the security of inside KGA by ensuring that each trapdoor corresponds to multiple keywords. Wang et al. [22] gave a PEKS scheme with dual servers. In 2017, Huang et al. [8] proposed the concept of public-key authentication searchable encryption. After that, Huang et al.'s scheme has been extended to certificateless PAEKS [23–25] and identity-based PAEKS [12]. And in the field of Internet of Things, many PEAKS variants [26–28] have been proposed. In 2019, Lu et al. [29] presented a PEKS scheme without random prediction. Later, Noroozi et al. [30] proposed that Huang et al.'s scheme is insecure in the case of multiple receivers.

In 2020, Qin et al. [9] presented a new PAEKS that is claimed to provide multi-ciphertext indistinguishability but no multi-trapdoor privacy. Recently, Li et al. [12] proposed a new PAEKS scheme under a designated server which still cannot guarantee MTP. Furthermore, almost PAEKS [8, 12] and their variants [9, 25, 31, 32] cannot provide MTP security and hide the search pattern of the user. Later, Qin et al. [33] proposed an improved security model and gave a specific scheme. Recently, Lattice-based searchable encryption schemes [34, 35] have been proposed which are claimed to guarantee stronger security.

Paper organization

The rest of this paper is organized as follows. In section 2, we introduce some preliminary knowledge. Then we review Li et al.'s scheme and give a security analysis for it in section 3. The fourth section defines the enhanced scheme and its security model. Section 5 gives a concrete construction scheme and proves that it satisfies the designed testability, MTP security and MCI security. Then in section 6, we compare and analyze our scheme with others. In the last section, we give a summary and a prospect for the future.

Preliminaries

Bilinear pairing

We briefly describe the definition of bilinear mapping. (See more details in [36]). Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ be a com-

putable bilinear pairing, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of prime order p. The map $\hat{e}$ has the following properties.

- For any $x, y \in \mathbb{Z}_p^*, g, g_1 \in \mathbb{G}_1$, the equation $\hat{e}\left(g^x, g_1^y\right) = \hat{e}\left(g, g_1\right)^{xy}$ holds.
- For any generator $g \in \mathbb{G}_1$, $\hat{e}(g,g)$ is a generator of $\mathbb{G}_2$.
- For any $g, g_1 \in \mathbb{G}_1$, there exists a PPT algorithm to compute $\hat{e}(g_1, g)$.

Complexity assumptions

In this subsection, $\mathbb{G}_1$ and $\mathbb{G}_2$ are two cyclic groups of prime order p, g is a generator of $\mathbb{G}_1$ and $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a bilinear map. Decisional Diffie–Hellman assumption and Decisional bilinear Diffie–Hellman assumption are introduced as follows.

Definition 1 *(Decisional Diffie–Hellman (DDH) assumption): Given g, g^x, $g^y \in \mathbb{G}_1$, where $x, y \in \mathbb{Z}_q^*$, there no exists polynomial-time algorithm to distinguish between (g, g^x, g^y, g^{xy}) and (g, g^x, g^y, Z), where $Z \in_R \mathbb{G}_1$. The advantage of adversary $\mathcal{A}$ is*

$$Adv_{\mathcal{A}}^{DDH}(\kappa) = |Pr[\mathcal{A}(g, g^x, g^y, g^{xy})] - Pr[\mathcal{A}(g, g^x, g^y, Z)]|$$

DDH assumption holds if the advantage is negligible.

Definition 2 *(Decisional bilinear Diffie–Hellman (DBDH) assumption) : Given g, g^x, g^y, $g^z \in \mathbb{G}_1$, where $x, y, z \in \mathbb{Z}_q^*$. The advantage of the adversary $\mathcal{A}$ is $Adv_{\mathcal{A}}^{DBDH}(\kappa) = |Pr\left[\mathcal{A}\left(g, g^x, g^y, g^z, e(g,g)^{xyz}\right)\right] - Pr\left[\mathcal{A}(g, g^x, g^y, g^z, Z\right]|$, where $x, y, z \in_R \mathbb{Z}_q^*$ and $Z \in_R \mathbb{G}_2$. DBDH assumption holds if the advantage is negligible.*

System model

Our system framework is showed in Fig. 1. The system contains three entities: a cloud server, a data owner and a receiver. Moreover, the data owner wants to send confidential files to the cloud which are allowed the assigned receiver to access the data. The exact procedures are as follows: First, the data owner extracts a group of keywords from documents and builds an secure index including keyword ciphertexts and documents. Second, the data owner encrypts the files by symmetric encryption and uploads the encrypted file and keyword ciphertext index to the server. Third, the receiver generates a trapdoor for a query keyword and sends it to the server. Finally, after receiving the trapdoor, cloud server runs the test algorithm and outputs the search results.In Table 1, we summarize the notations used in this paper.

Cryptanalysis of li et al.'s scheme

In this section, we review an identity-based searchable authenticated encryption scheme under a designated server proposed by Li et al.. After analyzing their scheme, we propose that it cannot guarantee trapdoor indistinguishability.

Review of li et al.'s scheme

Li et al.'s scheme consists of the following polynomial algorithms:

Setup(κ): From the security parameter κ, it outputs a public parameter $para = (\mathbb{G}_1, \mathbb{G}_2, \hat{e}, p, g, g_1, H, H_1, mpk)$ and msk, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic groups of prime order p. g and g_1 are generators of $\mathbb{G}_1$. $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is an efficient bilinear map, and $H : \mathbb{G}_2 \times \{0,1\}^* \rightarrow \mathbb{G}_1, H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$, $msk = \alpha \in \mathbb{Z}_p$, $mpk = g^\alpha$.

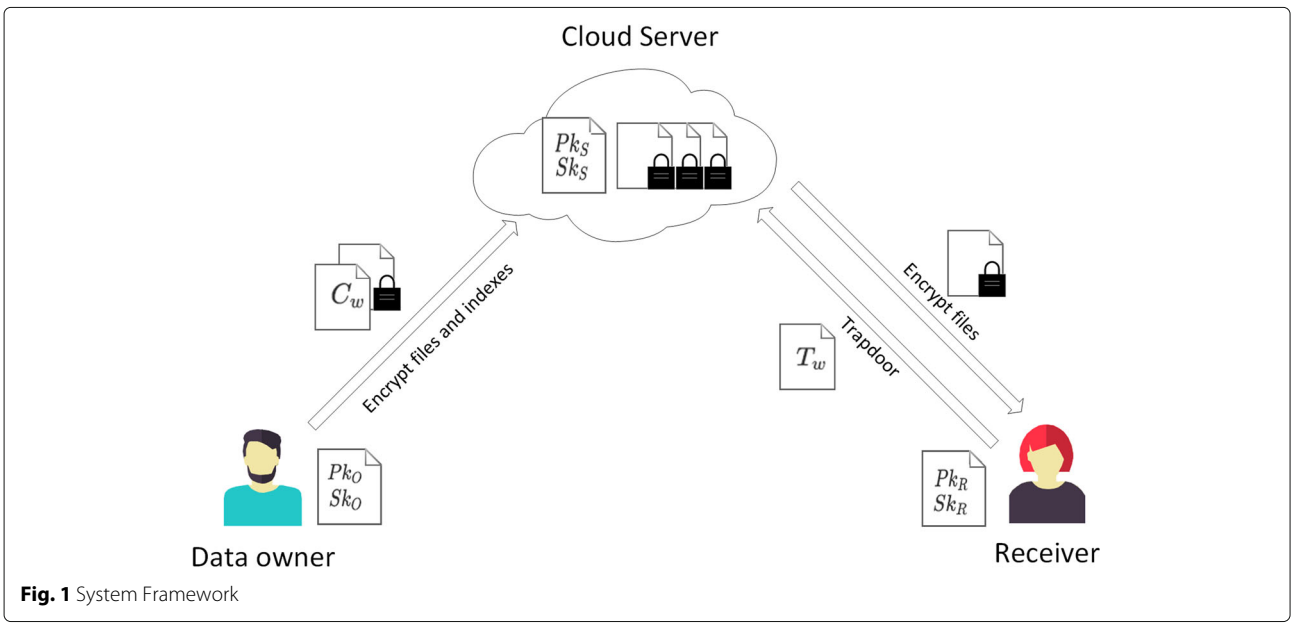

Fig. 1 System Framework

Table 1 Notations

Notation	Description
(Pk_O, Sk_O)	Data owner' keys
(Pk_R, Sk_R)	Receiver' keys
(Pk_S, Sk_S)	Server'keys
κ	Security parameter
C_w	Keyword ciphertext
$T_{w'}$	Search trapdoor
$\mathcal{C}, \mathcal{A}$	Challenger and adversary
$\mathcal{O}$	Oracles
g, h, g_1	Generators of group G_1
G_1, G_2	Multiplicative cyclic groups with order q
q	A large prime number
H_1	A hash function $H_1 : G_1 \rightarrow Z_q^*$
H_2	A hash function $H_2 : \{0,1\}^* \rightarrow Z_q^*$
pp	A global parameter
E	The time cost of one modular exponentiation
H_p	The time cost of one Hash-to-point operation
P	The time cost of one pairing operation
M	The time cost of one multiplication operation
H	The time cost of one hash function operation
$\lvert\mathbb{G}_1\rvert$	the bit length of an element in $\mathbb{G}_1$
$\lvert\mathbb{G}_2\rvert$	the bit length of an element in $\mathbb{G}_2$
$\lvert q\rvert$	the bit length of an element in $\mathbb{Z}_q$

$KGen_s(para)$: With the parameters $para$, it outputs the public/secret key pairs $(Pk_S, Sk_S) = \left(g^z, z\right)$ of the server, where $z \in_R \mathbb{Z}_p$.

$KGen_{usr}(pp, msk, ID)$: Inputting (pp, msk, ID), it returns $Sk_{ID} = H_1(ID)^{\alpha}$.

$dIBAEKS(para, w, Pk_S, Sk_{ID_O}, ID_O, ID_R)$: With $para$, w, Pk_s, Sk_{ID_O}, ID_O of a data owner and a receiver's ID_R, it returns a keyword ciphertext C_w=(C_1, C_2, C_3), where $C_1 = \hat{e}\left(H(k,w), Pk_S^s\right)$, $C_2 = g^s$, C_3=g_1^s, $s \in_R \mathbb{Z}_p$ and $k = \hat{e}(Sk_{ID_O}, H_1(ID_R))$.

$Tarpdoor(para, w, Pk_S, Sk_{ID_R}, ID_O, ID_R)$: It outputs a trapdoor $T_w = (H(k,w) \cdot g_1^r, g^r)$, where $k = \hat{e}(H_1(ID_O), Sk_{ID_R})$.

$Test(para, Sk_S, ID_O, ID_R, C_W, T_W)$: It outputs 1 if

$$C_1 \cdot \hat{e}\left(T_2^{Sk_S}, C_3\right) = \hat{e}\left(T_1^{Sk_S}, C_2\right),$$

and 0 otherwise.

Cryptanalysis of their scheme

In [12], Li et al. claimed that their dlBAEKS scheme satisfies the trapdoor indistinguishability under the random prediction model. Although trapdoor contains a random number in dlBAKES, there is an efficient algorithm to ascertain whether two trapdoors encrypt the identical keyword or not. In fact, for any two trapdoors $T_w = (T_1, T_2)$ and $T_{w'} = \left(T_1', T_2'\right)$ containing unknown keywords w and w', respectively, the decision algorithm by adversary is as follows:

$$\begin{aligned}
&\hat{e}(T_1, g) \cdot e\left(g_1, T_2\right)^{-1} \\
&= \hat{e}\left(g, H(k,w) \cdot g_1^r\right) \cdot e\left(g_1, g^r\right)^{-1} \\
&= \hat{e}(g, H(k,w)) e\left(g_1, g^r\right) e\left(g_1, g^r\right)^{-1} \\
&= \hat{e}(H(k,w), g)
\end{aligned}$$

where k, g are both fixed values for the same owner and receiver. An attacker captures some tuples $T_w = (T_1, T_2)$ and $T_{w'} = (T_1', T_2')$. This distinguishing attack works as follows: if

$$\hat{e}(T_1, g) \cdot \hat{e}(g_1, T_2)^{-1} = \hat{e}(T_1', g) \cdot \hat{e}(g_1, T_2')^{-1}$$

then $w = w'$, and $w \neq w'$ otherwise. Thus, the dIBAEKS scheme in [12] is insecure for multi-trapdoor privacy. This means that for the data owner sharing files to the receiver, the external attacker can effectively determine if multiple trapdoors generated by the receiver corresponds to the same keyword.

In addition, another scheme dIBAEKS-3 proposed in [12] also has the similar vulnerability. The decision algorithm is as follows: $\hat{e}(T_1, g_1) \cdot \hat{e}(T_2, g)^{-1} = \hat{e}(H(k,w), g_1)$. From two trapdoors: $T_w = (T_1, T_2)$ and $T_{w'} = (T_1', T_2')$, an attacker checks whether $\hat{e}(T_1, g_1) \cdot \hat{e}(T_2, g)^{-1} = \hat{e}(T_1', g_1) \cdot \hat{e}(T_2', g)^{-1}$ holds. If it holds, these two trapdoors are generated by the same keyword. Thus, the dIBAEKS-3 scheme is not able to provide multi-trapdoor privacy.

Definitions and security model

Definition

Our scheme consists of seven (probabilistic) polynomial-time(PPT) algorithms as follows.

- $Setup(\kappa) \rightarrow pp$: Given a security parameter κ, it returns the global parameter pp.
- $KeyGen_O(pp) \rightarrow (Pk_O, Sk_O)$: Given the parameter pp, it returns the public/secret key pairs(Pk_O, Sk_O) of the data owner.
- $KeyGen_R(pp) \rightarrow (Pk_R, Sk_R)$: With the parameter pp, it outputs the public/secret key pairs (Pk_R, Sk_R) of the receiver.
- $KeyGen_S(pp) \rightarrow (Pk_S, Sk_S)$: Inputting pp, it calculates the public key and private key pairs (Pk_S, Sk_S) of the server.
- $PEKS(pp, Pk_S, Pk_R, Sk_O, w) \rightarrow C_w$: Given the parameter pp, Pk_S of server, Pk_R of receiver, Sk_O of data owner and a keyword w, it outputs the ciphertext C_w.
- $Trapdoor(pp, Pk_O, Sk_R, w') \rightarrow T_{w'}$: With the parameter pp, the data owner's Pk_O, Pk_R of a receiver and a keyword w', it computes the trapdoor $T_{w'}$ of w'.

- *Test* $(pp, Sk_S, C_w, T_{w'}) \rightarrow \beta$: With pp, Sk_S, C_w and $T_{w'}$, it outputs 1 if C_w and $T_{w'}$ contain a same keyword, 0 otherwise.

Security model

In order to prevent an adversary obtaining any useful inforamtion of keywords, we define three games between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$, namely, multi-ciphertext indistinguishability, multi-trapdoor privacy and designated testability.

Game 1: Multi-ciphertext indistinguishability.

Setup: The challenger $\mathcal{C}$ runs $KeyGen_S$, $KeyGen_O$ and $KeyGen_R$ algorithms with pp to generate (Pk_S, Sk_S), (Pk_O, Sk_O) and (Pk_R, Sk_R). It returns the tuple $(pp, (Pk_S, Sk_S))$ to $\mathcal{A}$.

Phase 1: $\mathcal{A}$ can issue the following two oracles for polynomial number times.

- Ciphertext Oracle $\mathcal{O}_C$: With (Pk_O, Pk_R, Pk_S, w), $\mathcal{C}$ computes the ciphertext C_w and sends it to $\mathcal{A}$.
- Trapdoor Oracle $\mathcal{O}_T$: With (Pk_O, Pk_R, w), $\mathcal{C}$ computes a trapdoor T_w of a keyword w and returns it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ sends two tuples of challenge keywords $\vec{w}_0 = (w_{0,1}, \ldots, w_{0,n}), \vec{w}_1 = (w_{1,1}, \ldots, w_{1,n})$ to $\mathcal{C}$. However, the attacker cannot query the challenge keyword in tuple $\vec{w}_0$ or $\vec{w}_1$ in advance. $\mathcal{C}$ selects a random bit $b \in \{0, 1\}$, computes $C_{b,i} \leftarrow PEKS(pp, Pk_S, Pk_R, Sk_O, w_{b,i})$, and returns a ciphertext set $\vec{C}_b = (C_{b,1}, \ldots, C_{b,n})$ to the adversary $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ can continue to query $\mathcal{O}_C$ and/or $\mathcal{O}_T$ for any keyword w except $w \in \vec{w}_0 \cup \vec{w}_1$.

Guess: The adversary $\mathcal{A}$ sends its guess bit $\hat{b}'$ to $\mathcal{C}$. Therefore, the condition that $\mathcal{A}$ wins the game is $b' = b$. The advantage of any PPT attacker $\mathcal{A}$ who wins this game is defined as $Adv_{\mathcal{A}}^{MCI}(\kappa) = |Pr[b' = b] - \frac{1}{2}|$.

Game 2: Multi-trapdoor privacy.

Setup: Same as Game 1, $\mathcal{C}$ generates (Pk_S, Sk_S), (Pk_O, Sk_O) and (Pk_R, Sk_R) and gives $(pp, (Pk_S, Sk_S))$ to $\mathcal{A}$.

Phase 1: As in Game 1, an adversary can adaptively query the ciphertext oracle $\mathcal{O}_C$ and trapdoor oracle $\mathcal{O}_T$ in polynomial time.

Challenge: $\mathcal{A}$ sends two challenge keywords tuples $\vec{w}_0 = (w_{0,1}, \ldots, w_{0,n})$, $\vec{w}_1 = (w_{1,1}, \ldots, w_{1,n})$ to $\mathcal{C}$. However, the attacker cannot query the challenge key in tuple $\vec{w}_0$ or $\vec{w}_1$ in advance. $\mathcal{C}$ computes and returns a trapdoor set $\vec{T}_b = (T_{b,1}, \ldots, T_{b,n})$ of a random bit $b \in \{0, 1\}$.

Phase 2: As in Phase 1, $\mathcal{A}$ can continue to query $\mathcal{O}_C$ and/or $\mathcal{O}_T$ for any keyword w except $w \in \vec{w}_0 \cup \vec{w}_1$.

Guess: $\mathcal{A}$ sends its guess bit $\hat{b}'$ to $\mathcal{C}$. Therefore, $\mathcal{A}$ will win the game if $b' = b$. The advantage of all PPT adversaries $\mathcal{A}$ who win the game is defined as $Adv_{\mathcal{A}}^{MTP}(\kappa) = |Pr[b' = b] - \frac{1}{2}|$.

Game 3: Designated testability.

$\mathcal{A}$ is an external adversary who can get the keyword ciphertext and the trapdoor by monitoring the public channel. However, $\mathcal{A}$ cannot get the secret key of the server. Designated testability ensures that only a designated server who own the private key can search a keyword over ciphertexts.

Setup: $\mathcal{C}$ runs $KeyGen_S$, $KeyGen_O$ and $KeyGen_R$ algorithms with pp to generate the public and private key pairs (Pk_S, Sk_S), (Pk_O, Sk_O) and (Pk_R, Sk_R). It then sends the tuple (pp, Pk_S) to $\mathcal{A}$.

Phase 1: There are two oracles as follows, which allow $\mathcal{A}$ to query in polynomial time.

- Ciphertext Oracle $\mathcal{O}_C$: With (Pk_O, Pk_R, Pk_S, w), $\mathcal{C}$ computes and returns the ciphertext C_w.
- Trapdoor Oracle $\mathcal{O}_T$: Input a tuple (Pk_O, Pk_R, w), $\mathcal{C}$ computes and outputs trapdoor T_w.

Challenge: $\mathcal{A}$ sends two challenge keywords w_0, w_1 to $\mathcal{C}$, then $\mathcal{C}$ calculates and outputs C_b of a random bit $b \in \{0, 1\}$.

Phase 2: As in Phase 1, $\mathcal{A}$ can carry on querying for any keyword w_i except $w_i \in (w_0, w_1)$.

Guess: The adversary $\mathcal{A}$ sends its guess bit $\hat{b}'$ to $\mathcal{C}$. Therefore, $\mathcal{A}$ wins the game if $b' = b$. The advantage for all PPT attackers who win the game is defined as $Adv_{\mathcal{A}}^{DT}(\kappa) = |Pr[b' = b] - \frac{1}{2}|$.

Proposed scheme

Construction

In this section, we propose a concrete construction of our scheme that can provide multi-ciphertext indistinguishability, multi-trapdoor privacy and security against key guessing attack. The details of proposed scheme are described as follows.

- $Setup(\kappa)$: From κ, it chooses a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1, \mathbb{G}_2$ are cyclic groups of prime order q, and selects two random generators $g, h \in \mathbb{G}_1$ and two cryptographic hash functions H_1: $\mathbb{G}_1 \rightarrow \mathbb{Z}_q^*$, H_2: $\{0, 1\}^* \rightarrow \mathbb{Z}_q^*$. It returns the public parameter $pp = (\mathbb{G}_1, \mathbb{G}_2, q, g, h, \hat{e}, H_1, H_2)$.
- $KeyGen_O(pp)$: It takes a grobal pubilc parameter pp as inputs, selects $x \leftarrow Z_p$ randomly and defines $Pk_O = g^x$ and $Sk_O = x$. It outputs the data owner's public/secret key pairs (Pk_O, Sk_O).
- $KeyGen_R(pp)$: From pp, it chooses a random $y \leftarrow Z_p$ and sets $Pk_R = g^y$ and $Sk_R = y$ then returns the receiver's public/secret key pairs (Pk_R, Sk_R).
- $KeyGen_S(pp)$: By a grobal pubilc parameter pp, it selects randomly $z \leftarrow Z_p$, and defines $Pk_S = h^z$ and $Sk_S = z$. Finally it returns the server's public/secret key pairs (Pk_S, Sk_S).

- *PEKS*$(pp, Pk_S, Pk_R, Sk_O, w)$: Given the public parameter pp, Pk_S, Pk_R, Sk_O and a keyword w, a data owner performs the following steps:
  - Select a number $r \in_R \mathbb{Z}_q^*$.
  - Calculate $C_1 = h^r, C_2 = \hat{e}(Pk_R, Pk_S)^{rSk_O H_2(w)}$ and $C_3 = g^{rk}$, where $k = H_1\left(Pk_R^{Sk_O}\right)$.
  - Output the ciphertext $C_w = (C_1, C_2, C_3)$ of w.
- *Trapdoor*(pp, Pk_O, Sk_R, w'): From pp, Pk_O of data owner, Sk_R of receiver and a keyword w', a receiver executes the following steps:
  - Choose a number $s \in_R \mathbb{Z}_q^*$.
  - Compute $T_1 = {Pk_S}^s$ and $T_2 = {Pk_O}^{Sk_R H_2(w')} \cdot g^{sk}$, where $k = H_1\left(Pk_O^{Sk_R}\right)$.
  - Return the trapdoor $T_{w'} = (T_1, T_2)$
- *Test*(pp, Sk_S, C_w, T_w): After receiving $T_{w'}$, the server searchs over keyword ciphertexts $\{C_w\}$ by testing $\hat{e}(T_2, C_1^{Sk_S}) = \hat{e}(T_1, C_3) \cdot C_2$ using his private key Sk_S. If the equation holds, it outputs 1; otherwise, it outputs 0.

Correctness: Assume that (PK_O, Sk_O), (PK_R, Sk_R) and (PK_S, Sk_S) be the data owner, the receiver and the server's public/secret key pairs respectively. $C_w = (C_1, C_2, C_3)$ is the ciphertext of a keyword w generated by the owner. $T_{w'} = (T_1, T_2)$ is a trapdoor of a keyword generated by the receiver. It follows that:

$$\begin{aligned}\hat{e}(T_2, C_1^{Sk_S}) &= \hat{e}\left({Pk_O}^{Sk_R H_2(w')+rk}, h^{Sk_S r}\right)\\ &= \hat{e}(g,h)^{rxyzH_2(w')} \cdot \hat{e}(g,h)^{rszk}.\\ \hat{e}(T_1, C_3) \cdot C_2 &= \hat{e}\left(h^{zs}, g^{rk}\right) \cdot \hat{e}\left(g^y, h^z\right)^{rxH_2(w)}\\ &= \hat{e}(g,h)^{rxyzH_2(w)} \cdot \hat{e}(g,h)^{rszk}.\end{aligned}$$

Thus, if $w = w'$, then $\hat{e}\left(T_2, C_1^{Sk_S}\right) = \hat{e}(T_1, C_3) \cdot C_2$ holds with probability 1; otherwise, it holds with overwhelming probability by the collision resistance of the hash function H_2.

Security proof

In this subsection, we prove that our scheme achieves the security of MCI, MTP and designated testability. Formally, we have the following theorems.

Theorem 1 *Under the assumption of DBDH, our scheme satisfies multi-ciphertext indistinguishability.*

Proof Assume that $\mathcal{A}$ is an adversary who tries to destroy the MCI security. And the algorithm $\mathcal{C}$ for solving DBDH problem is established. Given a instance of this problem, such as $Y = \left(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, g, g^x, g^y, g^z, Z_1\right)$, the algorithm $\mathcal{C}$ works exactly as follows.

Setup: $\mathcal{C}$ randomly selects two hash functions $H_1 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ and sets $pp = \left(\mathbb{G}_1, \mathbb{G}_2, q, g, \hat{e}, h = g^\alpha, H_1, H_2\right)$, $Pk_O = g^x$, $Pk_R = g^y$ and $(Pk_S, Sk_S) = \left(h^t, t\right)$. It then sends pp and (Pk_S, Sk_S) to $\mathcal{A}$.

Phase 1: We define several oracles as follows, which allow $\mathcal{A}$ to query many times. We assume that $\mathcal{A}$ cannot query the same oracle more than once.

- Hash Oracle $\mathcal{O}_{H_1}$: In response to the H_1 query, the oracle maintains a tuple list $L_{H_1} = \{< m_i, a_i >\}$. We assume that $\mathcal{O}_{H_1}$ can be asked by attackers for q_{H_1} times at most. For querying m_i to the oracle, it will perform the following operations: At first, if $\hat{e}(g, m_i) = \hat{e}\left(g^x, g^y\right)$, $\mathcal{C}$ randomly returns a bit b' and halts. Otherwise $\mathcal{C}$ checks whether m_i exists in the tuple list. If so, $\mathcal{C}$ takes out the corresponding tuple and returns a_i to $\mathcal{A}$. Otherwise, it randomly chooses a new exponent $a_i \in \{0,1\}^\kappa$, stores $< m_i, a_i >$ in L_{H_1} and returns a_i to $\mathcal{A}$.
- Hash Oracle $\mathcal{O}_{H_2}$: In response to the H_2 query, the oracle maintains a tuple list $L_{H_2} = \{< w_i, b_i >\}$. We assume that $\mathcal{O}_{H_2}$ can be asked by attackers for q_{H_2} times at most. When submitting the keywords w_i to the Oracle for query, it will perform the following operations: At first, it checks whether w_i exists in the tuple list. if it exists, $\mathcal{C}$ will take out the corresponding tuple and return b_i to $\mathcal{A}$. Otherwise, it randomly selects a new exponent $b_i \in \{0,1\}^\kappa$, stores $< w_i, b_i >$ in L_{H_2} and returns b_i to $\mathcal{A}$.
- Oracle $\mathcal{O}_E$: It takes public key Pk_i as input. To response to the queries, the oracle maintains a tuple list $L_E = \{< Pk_i, c_i, V_i >\}$, and it is assumed that $\mathcal{O}_E$ can be asked by attackers for q_E times at most. When submitting Pk_i to the Oracle query, it will perform the following operations: At first, if $Pk_i = g^x$ or $Pk_i = g^y$, $\mathcal{C}$ randomly returns a bit b' and halts. Otherwise $\mathcal{C}$ tests whether exists Pk_i in the tuple list. If so, $\mathcal{C}$ chooses the candidate tuple and returns c_i to $\mathcal{A}$. Otherwise, it randomly selects a new exponent $c_i \in \{0,1\}^\kappa$, and computes $V_i = {Pk_i}^{c_i}$. Finally, it stores $< Pk_i, c_i, V_i >$ in L_E and outputs c_i.
- Ciphertext Oracle $\mathcal{O}_{Ciphertext}$: Input a tuple (Pk_O, Pk_R, w_i), which $w_i \in \{0,1\}^*$, it randomly chooses $r_i \in \mathbb{Z}_q^*$, and computes $C_{w_i} = \left(C_{1w_i}, C_{2w_i}, C_{2w_i}\right)$ as follows.
  - If $(Pk_O, Pk_R) = \left(g^x, g^y\right)$ or $(Pk_O, Pk_R) = \left(g^y, g^x\right)$, then it sets $g^z = g^{\alpha r_i}$, and computes $C_{1w_i} = g^z$, $C_{2w_i} = Z_1^{tb_i}$, $C_{3w_i} = g^{r_i a_i}$.
  - Otherwise, at least one Pk_i in (Pk_O, Pk_R) is equal to g^x or g^y. It computes $H_2(w_i) = b_i$,

$k = a_i$, and returns to $\mathcal{A}$ with $C_{w_i} = \left(C_{1w_i}, C_{2w_i}, C_{3w_i}\right)$, where $C_{1w_i} = h^{r_i}$, $C_{2w_i} = \hat{e}(g^y, h^{r_i})^{tc_o b_i}$ and $C_{3w_i} = g^{r_i a_i}$.

- Trapdoor Oracle $\mathcal{O}_{Trapdoor}$: Input (Pk_O, Pk_R, w_i), where $w_i \in \{0,1\}^*$, it randomly chooses $s_i \in \mathbb{Z}_q^*$, and computes $T_{w_i} = (T_{1w_i}, T_{2w_i})$ as follows.
  - If $(Pk_O, Pk_R) = (g^x, g^y)$ or $(Pk_O, Pk_R) = (g^y, g^x)$, then it computes $T_{1w_i} = g^{ts_i}$ and $T_{2w_j i} = Z_2^{b_i} \cdot g^{r_i a_i}$.
  - Otherwise, at least one Pk_i in (Pk_O, Pk_R) equals to g^x or g^y. It calculates $H_2(w_i) = b_i, k = a_i$, and returns to $\mathcal{A}$ with $T_{w_i} = \left(T_{1w_i}, T_{2w_i}\right)$, where $T_{1w_i} = g^{ts_i}$, $T_{2w_i} = (g^x)^{c_o b_i} \cdot g^{s_i a_i}$.

Challenge: $\mathcal{A}$ completes multiple queries on the above oracles. It selects two challenge keyword tuples $\vec{w}_0^*$ and $\vec{w}_1^*$, and sends them to $\mathcal{C}$ with Pk_O^* and Pk_R^*. $\mathcal{C}$ randomly selects a random number r_i and a bit $\hat{b} \in \{0,1\}$. then $\mathcal{C}$ outputs a ciphertext tuple $\vec{C}_{w_{\hat{b}}{}^*} = \left(C_{w_{\hat{b},1}{}^*}, \ldots, C_{w_{\hat{b},n}{}^*}\right)$ where $C_{1w_{\hat{b},i}{}^*} = g^z$, $C_{2w_{\hat{b},i}{}^*} = Z_1^{tb_i}$, $C_{3w_{\hat{b},i}{}^*} = g^{za_i}$.

Phase 2: As with Phase 1 of operation, $\mathcal{A}$ continue to enquire $\mathcal{O}_{Ciphertext}$ and/or $\mathcal{O}_{Trapdoor}$ for any keyword w_i except $w_i \in \vec{w}_0 \cup \vec{w}_1$.

Guess: The adversary$\mathcal{A}$ sends its guess bit $\hat{b}'$ to $\mathcal{C}$. Returns $b' = 0$ if $\hat{b}' = \hat{b}$, $b' = 1$ otherwise.

If the guess of the challenging public key is incorrect, $\mathcal{C}$ will abort. This event is represented by E. If $\mathcal{C}$ aborts, $\mathcal{C}$ outputs a random bit. The termination probability of E is $\frac{1}{q_E(q_E-1)}$, therefore, $Pr[\overline{E}] = \frac{1}{q_E(q_E-1)}$.

Assume that algorithm $\mathcal{C}$ is not aborted. If the simulation provided by algorithm $\mathcal{C}$ is the same as scenario of $\mathcal{A}$ in real attack and $Z_1 = \hat{e}(g,g)^{xyz}$, the adversary $\mathcal{A}$ will win with $Adv_{\mathcal{A}}^{MCI}(\kappa) + \frac{1}{2}$. If Z_1 is randomly chosen from the group $\mathbb{G}_2$, $C_{2w_{\hat{b},i}{}^*} = Z_1^{Sk_S H_2(w)}$ is a random element of $\mathbb{G}_2$. In this case, the trapdoor $\vec{T}_{w_{\hat{b}}{}^*}$ and ciphertext $\vec{C}_{w_{\hat{b}}{}^*}$ can be tested. When the keywords are consistent, test algorithm outputs 1. Therefore, $\mathcal{A}$ has a 1/2 probability that he wins the Game 1. Thus, the advantage for $\mathcal{C}$ in solving DBDH problem is

$$
\begin{aligned}
&Adv_{\mathcal{B}}^{DBDH}(\kappa) \\
&= |Pr[b' = b \mid E] Pr[E] + Pr[b' = b \mid \overline{E}] \cdot Pr[\overline{E}] - \frac{1}{2}| \\
&= |\frac{1}{2} \cdot (1 - Pr[\overline{E}]) + \left(Pr[b' = 0] \mid \overline{E} \cap b = 0\right) \cdot Pr[b = 0] \\
&\quad + Pr[b' = 1 \mid \overline{E} \cap b = 1] \cdot Pr[\overline{E}] - \frac{1}{2}| \\
&\geq |Pr[\overline{E}] \cdot \left(\left(Adv_{\mathcal{A}}^{MCI} + \frac{1}{2}\right) \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2}\right) - \frac{1}{2} \\
&\quad + \frac{1}{2} \cdot (1 - Pr[\overline{E}]) + Pr[\overline{E}] | \\
&= \frac{1}{2} Pr[\overline{E}] \, Adv_{\mathcal{A}}^{MCI}(\kappa) \\
&= \frac{1}{2q_E(q_E-1)} \cdot Adv_{\mathcal{A}}^{MCI}(\kappa).
\end{aligned}
$$

Theorem 2 *Under the assumption of DDH, our scheme satisfies semantically MTP security.*

Proof Assume that $\mathcal{A}$ is an external opponent who tries to crack the Multi-trapdoor Privacy. Moreover, the algorithm $\mathcal{C}$ for solving the DDH problem is established. Given a instance of this problem, such as $Y = \left(\mathbb{G}_1, q, g, g^x, g^y, Z_2\right)$, the algorithm $\mathcal{C}$ works exactly as follows.

Setup: $\mathcal{C}$ randomly selects two hash functions $H_1 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ and sets $pp = \left(\mathbb{G}_1, \mathbb{G}_2, q, g, h = g^\alpha, H_1, H_2\right)$, $Pk_O = g^x$, $Pk_R = g^y$ and $(Pk_S, Sk_S) = \left(h^t, t\right)$. It then sends pp to $\mathcal{A}$.

Phase 1: Same as in Theorem 1.

Challenge: $\mathcal{A}$ completes multiple queries on the above research. It selects two challenge keyword tuples $\vec{w}_0^*$ and $\vec{w}_1^*$, and sends them to $\mathcal{C}$ with Pk_O^* and Pk_R^*. $\mathcal{C}$ randomly select a number s_i and a bit $\hat{b} \in \{0,1\}$. Then, $\mathcal{C}$ returns a trapdoor set $\vec{T}_{w_{\hat{b}}{}^*} = \left(T_{w_{\hat{b},1}{}^*}, \ldots, T_{w_{\hat{b},n}{}^*}\right)$ where $T_{1w_{\hat{b},i}{}^*} = h^{ts_i}$, $T_{2w_{\hat{b},i}{}^*} = Z_2^{b_i} \cdot g^{s_i a_i}$.

Phase 2: $\mathcal{A}$ continue to enquire $\mathcal{O}_C$ and/or $\mathcal{O}_T$ for any keyword w_i except $w_i \in \vec{w}_0 \cup \vec{w}_1$.

Guess: The adversary $\mathcal{A}$ sends its guess bit $\hat{b}'$ to $\mathcal{C}$. He returns $b' = 0$ if $\hat{b}' = \hat{b}$, $b' = 1$ otherwise.

If the guess of challenging public key is incorrect, $\mathcal{C}$ will abort. This event will be represented by E. If $\mathcal{B}$ aborts, $\mathcal{C}$ outputs a random bit. The probability that it being equal to b is 1/2. According to the random guess of b, the termination probability of E is $\frac{1}{q_E(q_E-1)}$, therefore, $Pr[\overline{E}] = \frac{1}{q_E(q_E-1)}$.

Othervise, $\mathcal{C}$ does not abort. If the simulation provided by algorithm $\mathcal{C}$ is the same as scenario of $\mathcal{A}$ in real attack and $Z_2 = g^{xy}$, an adversary $\mathcal{A}$ will win the game with the probability of $Adv_{\mathcal{A}}^{MTP}(\kappa) + \frac{1}{2}$. If Z_2 is randomly chosen from the group $\mathbb{G}_2$, $T_{2w_{\hat{b},i}{}^*} = Z_2^{b_i} \cdot g^{s_i a_i}$ will be a random element of $\mathbb{G}_2$. Therefore, the challenge trapdoor tuple hides $\hat{b}$ completely. In this case, the adversary can test the trapdoor $\vec{T}_{w_{\hat{b}}{}^*}$ and the ciphertext $\vec{C}_{w_{\hat{b}}{}^*}$. When the keywords are equal, the test algorithm outputs 1. Thus, the advantage for $\mathcal{C}$ in solving DDH problem is equal to the advantage in Theorem 1.

Theorem 3 *Under the assumption of DBDH, our scheme satisfies designated testability.*

Table 2 Symbols and execution times(ms)

Notation	Description	Times
E	The time cost of one modular exponentiation	0.784
H_p	The time cost of one Hash-to-point operation	5.679
P	The time cost of one pairing operation	5.429
M	The time cost of one multiplication operation	2.108
H	The time cost of one hash function operation	0.008
$\|\mathbb{G}_1\|$	the bit length of an element in $\mathbb{G}_1$	-
$\|\mathbb{G}_2\|$	the bit length of an element in $\mathbb{G}_2$	-
$\|q\|$	the bit length of an element in $\mathbb{Z}_q$	-

Proof Assume that $\mathcal{A}$ is an attacker who tries to crack the designated testability and the challenger $\mathcal{C}$ wants to solve DBDH problem. Given a instance of this problem, such as $Y = \left(\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, h, h^x, h^y, h^z, Z_3\right)$, the algorithm $\mathcal{C}$ works as follows.

Setup: $\mathcal{C}$ randomly selects two hash functions $H_1 : \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$ and sets $pp = (\mathbb{G}_1, \mathbb{G}_2, q, \hat{e}, h, H_1, H_2, g = h^x)$, $(Pk_O, Sk_O) = \left(g^s, s\right)$, $(Pk_R, Sk_R) = (g^t, t)$ and $Pk_S = h^z$. It then sends pp to $\mathcal{A}$.

Phase 1: Hash Oracle $\mathcal{O}_{H_1}$ and Hash Oracle $\mathcal{O}_{H_2}$ are same in Theorem 1. We only define Exact Oracle $\mathcal{O}_E$ as follows.

- Exact Oracle $\mathcal{O}_E$: Given Pk expect Pk_S, the algorithm $\mathcal{C}$ returns Sk.

Frist $\mathcal{A}$ performs multiple queries on the above oracle. It selects two challenge keywords w_0^* and w_1^*. Then $\mathcal{A}$ returns (w_0^*, w_1^*) to $\mathcal{C}$ together with Pk_O^* and Pk_R^*. $\mathcal{C}$ selects a number $y \in_R \mathbb{Z}_q$ and a bit $\hat{b} \in_R \{0,1\}$, and outputs $C_{w_{\hat{b}}^*} = \left(C_{1w_{\hat{b}}^*}, C_{2w_{\hat{b}}^*}, C_{3w_{\hat{b}}^*}\right)$ where $C_{1w_{\hat{b}}^*} = h^y$, $C_{2w_{\hat{b}}^*} = Z_3^{stb_i}$, $C_{3w_{\hat{b}}^*} = g^{yk}$.

Phase 2: $\mathcal{A}$ continue to enquire $\mathcal{O}_C$ and/or $\mathcal{O}_T$ for any keyword w_i except $w_i \in \{w_0, w_1\}$.

Guess: The adversary $\mathcal{A}$ transmits its guess bit $\hat{b}'$ to $\mathcal{C}$. Returns $b' = 0$ if $\hat{b}' = \hat{b}$, $b' = 1$ otherwise. If the simulation provided by algorithm $\mathcal{C}$ is the same as $\mathcal{A}$ in real attack and $Z = \hat{e}(h,h)^{xyz}$, $\mathcal{A}$ will win the game with the probability of $Adv_{\mathcal{A}}^{DT}(\kappa) + \frac{1}{2}$. If Z is randomly chosen from the group $\mathbb{G}_2$, $C_{2w_{\hat{b},i}^*} = Z^{Sk_S H_2(w)}$ is a well distributed challenge ciphertext. And $\mathcal{A}$ has a 1/2 probability of winning the game. Thus, $\mathcal{C}$'s advantage in solving DBDH problem is

$$
\begin{aligned}
&Adv_{\mathcal{B}}^{DBDH}(\kappa) \\
&= |Pr[\,b' = 1 \mid b = 1] \cdot Pr[\,b = 1] \\
&\quad + Pr[\,b' = 1 \mid b = 1] \cdot Pr[\,b = 1] - \frac{1}{2}| \\
&= |\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \left(Adv_{\mathcal{A}}^{DT}(\kappa) + \frac{1}{2}\right) - \frac{1}{2}| \\
&= \frac{1}{2} Adv_{\mathcal{A}}^{DT}(\kappa).
\end{aligned}
$$

Perfomance analysis

In this section, we analyze the performance of our scheme and the existing schemes(PAKES scheme [8], dIBAEKS scheme [12], SCF-PEKS scheme [3], dPEKS scheme [4] and Pan-Li's scheme [10]) in terms of computational and communication overheads. Moreover, we analyze the security between these PEKS schemes in MCI, MTP and security against KGA.

To evaluate the efficiency, we implemented the operations in our schemes using the MRACL library [37] on a personal notebook computer with an I7-8750H 2.20GHz processor, 16 GB memory, and Window 10 operating system.

First, we give the elapsed time of main operations used in searchable encryption schemes in Table 2. Main operations are pairing operation P, Hash-to-point operation H_p, modular exponentiation E and multiplication operation M in G_1, where $P \approx H_p > M > E \gg H$. The general hash operation takes less time than the above operations in Table 2. Thus, it is ignored in our computation analysis.

From Table 3, we give a theoretical efficiency comparison in computational time and communication complexity of PEKS algorithm, Trapdoor algorithm, and Test algorithm of our scheme and previous schemes [3, 4, 8, 10, 12]. In terms of computational efficiency, compared with

Table 3 Computation and Communcitaion efficiency comparison

Scheme	Computation			Communcitaion		
	PEKS	Trapdoor	Test	$\|C\|$	$\|T\|$	$\|Pk\|$
Huang and Li [8]	$H_p + 3E$	$P + H_P + E$	$2P + M$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|$
Li et al. [12]	$2P + 2H_p + 3E + H$	$P + 2H_p + 2E + M$	$2P + 2E + M$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_2\|$	$2\mathbb{G}_1$	*
Baek et al. [3]	$2P + H_p + 2E + M + H$	$H_p + E$	$P + M + E + H$	$\|\mathbb{G}_1\|+\|q\|$	$\|\mathbb{G}_2\|$	$2\|\mathbb{G}_1\|$
Rhee et al. [4]	$P + H_p + 2E + H$	$H_p + 3E + M$	$P + H_p + 2E$	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$
Pan and Li [10]	$H_p + 3E + H$	$P + H_p + 3E$	$2P + M$	$2\,\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|$
Our Scheme	$P + 4E + 2H$	$4E + 2H + M$	$2P + M + E$	$2\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$

other algorithms, PEKS and trapdoor algorithms are more efficient without using hash-to-point operation. Among the Test algorithms, our scheme is slightly weaker than other schemes, because it adds the designated testability to ensure that only the specified server can perform search operations. In terms of communication efficiency, our efficiency is basically the same as other schemes. Figs. 2, 3 and 4 demonstrate the practical performance of PEKS algorithm, Trapdoor algorithm, and Test algorithm, respectively.

As shown in Fig. 2, the computation cost to encrypt the keywords is lower than the three schemes [3, 4, 12] and is similar to that of Huang et al.'s scheme [8] and Pan and Li's scheme [10]. For the efficiency of trapdoor algorithm, Fig. 3 illustrates that the trapdoor algorithm in our scheme runs much faster than that in all schemes [3, 4, 8, 10, 12] because our trapdoor algorithm performs no pairing and Hash-to-point operations. In Fig. 4, the computation complexity in our scheme is higher than Baek et al.'s scheme [3] and is not worse than other four schemes. To ensure that the user-side algorithms (Trapdoor and PEKS algorithms) have higher security and efficiency, we add the server's private key to the test algorithm for stronger security, thus the efficiency of the server's Test algorithm is compromised.

Moreover, Table 4 illustrates the security comparison including MCI security, MTP security, Inside KGA, and Requirement for the secure channel between our scheme and these existing schemes. As shown in Table 4, Huang et al.' s scheme [8] can resist inside KGA, but it needs a secure channel and cannot provide MCI security. Li et al.'s scheme [12], Baek et al.'s scheme [3] and Rhee et al.'s scheme [4] can provide MCI security but not guarantee MTP and security against KGA. Pan and Li's scheme [10] is able to resist inside KGA and have MTP security but cannot satisfy MCI security. Our scheme satisfies MCI, MTP and security against inside KGA.

Conclusion

In this paper, we first analyze the security of Li et al.'s scheme and propose a multi-trapdoor attack against it. Next, we construct a secure public-key searchable encryption scheme with designated server based on Diffie-Hellman problem. It is proved that our scheme can provide multi-ciphertext indistinguishability, multi-trapdoor privacy security and designated testability. Then we compare our scheme with others in terms of communication cost and computational cost. The results show that our scheme is more efficient in keyword ciphertext and trapdoor algorithms. However, our scheme can not prevent the server from executing the multi-trapdoor attack since the server can construct a certain equation by his private key to obtain the relationship of multiple trapdoors. As our future work, we will explore achieving multi-trapdoor privacy of keywords for the inside servers.

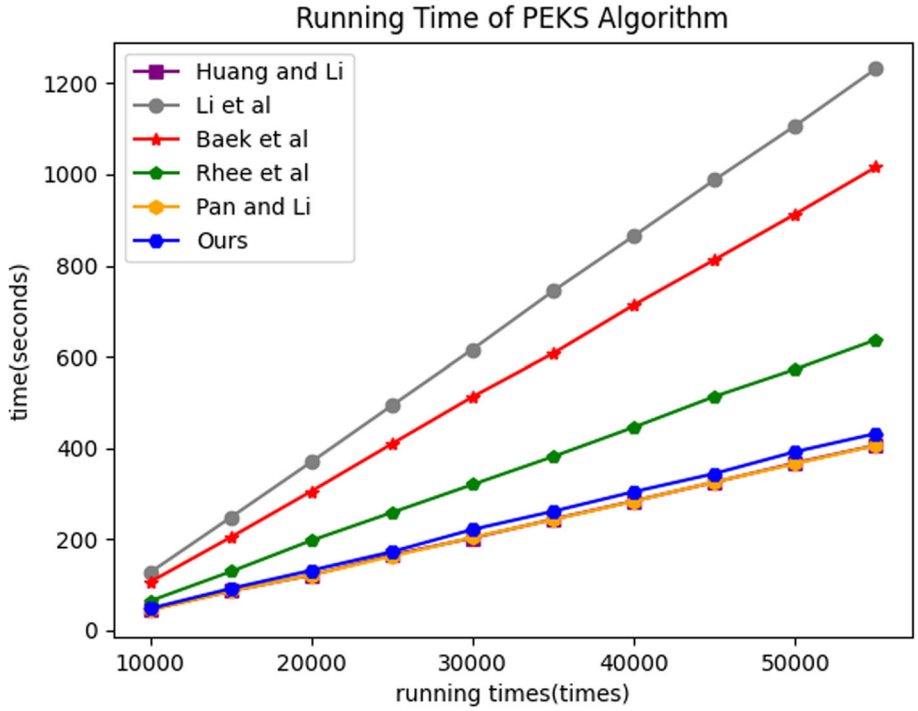

Fig. 2 Running Time of PEKS Algorithm

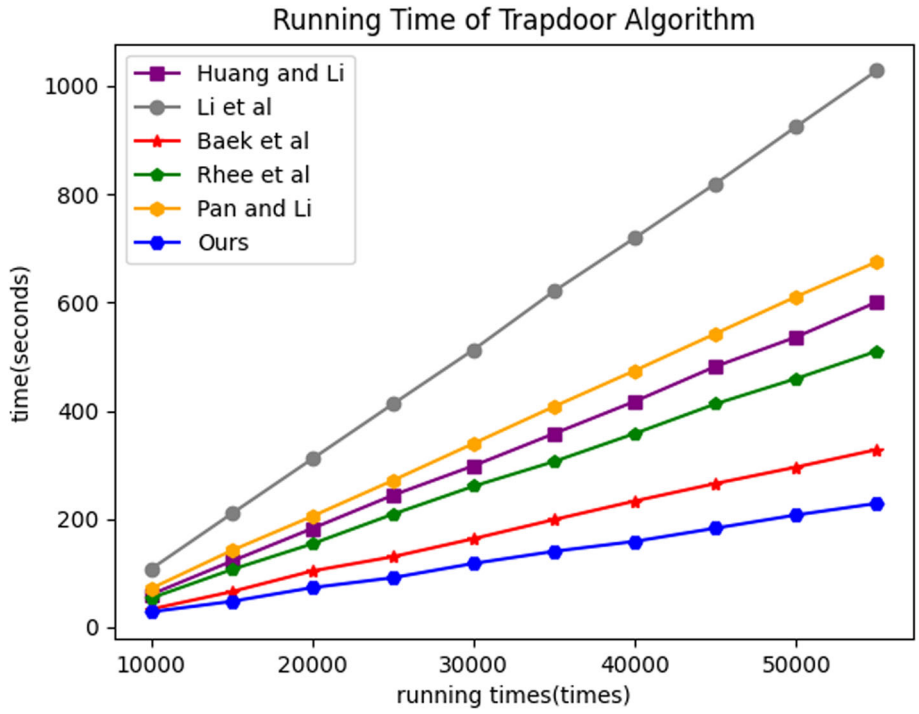

Fig. 3 Running Time of Trapdoor Algorithm

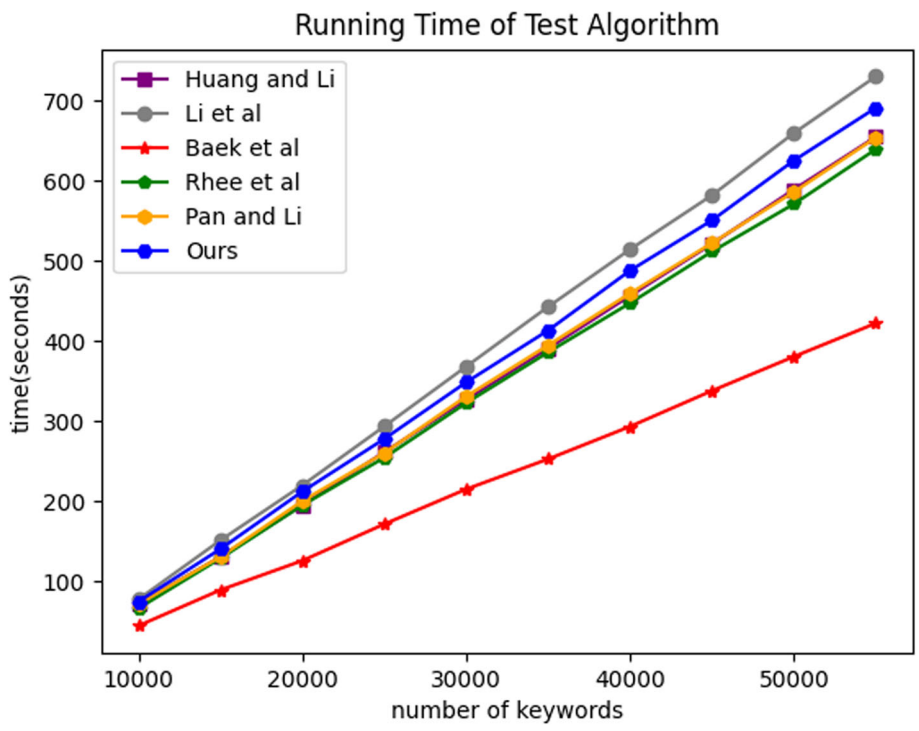

Fig. 4 Running Time of Test Algorithm

Table 4 Security comparison

Scheme	MCI-security	MTP-security	Inside KGA	Secure channel
Huang and Li [8]	No	No	Yes	Yes
Li et al. [12]	Yes	No	No	No
Baek et al. [3]	Yes	No	No	No
Rhee et al. [4]	Yes	No	No	No
Pan and Li [10]	No	Yes	Yes	No
Our Scheme	Yes	Yes	Yes	No

Acknowledgements
The authors would like to thank the anonymous reviewers for their insightful comments and suggestions on improving this paper.

Authors' contributions
This research paper was completed by the joint efforts of five authors. Therefore, any author participates in every part of the paper. But the basic roles of each author are summarized as follows: J.G. is the designer of the proposed model and method. L.H. is the corresponding author and the coordinator of the group, assisting J.G. in model design. G.Y. is the implementer and tester of the algorithm. X.L. is the main reviewer of this paper. C.T. is responsible for the experiment of the proposed method. All authors have read and agreed to the published version of the manuscript.

Authors' information
Junling Guo is a postgraduate in Cyberspace Security at School of Information Science and Technology, Hangzhou Normal University, China. His research interests include searchable encryption and public key cryptography.

Lidong Han received his Ph.D. degree from school of mathematics in Shandong university in 2010. Currently, he is working at Key Laboratory of Cryptography Technology of Zhejiang Province, and School of Information Science and Technology, Hangzhou Normal University. His research interests include cryptography, cloud computing, and remote user authentication.

Guang Yang is a postgraduate in Cyberspace Security at School of Information Science and Technology, Hangzhou Normal University, China. His research interests include data integrity, searchable encryption and public-key cryptography.

Xuejiao Liu received the BS, MS and PhD degrees in computer science from Huazhong Normal University, Wuhan, China. Now she is an associate professor in Hangzhou Normal University, Hangzhou, China. Her research interests cover network security, cloud security, security of Internet of vehicle and etc.

Chengliang Tian received the B.S. and M.S. degrees in mathematics from Northwest University, Xi'an, China, in 2006 and 2009, respectively, and the Ph.D. degree in information security from Shandong University, Jinan, China, in 2013. He held a postdoctoral position with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing. He is currently with the College of Computer Science and Technology, Qingdao University, as an Associate Professor. His research interests include latticebased cryptography, cloud computing security and privacy-preserving technology.

Funding
This work is supported by the National Natural Science Foundation of China(No.61702153, No.61972124) and the Natural Science Foundation of Zhejiang Province (No.LY19F020021).

Availability of data and materials
The datasets used during the current study are available from the corresponding author on reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Information Science and Technology, Hangzhou Normal University, Hangzhou, China. [2]Key Laboratory of Cryptography Technology of Zhejiang Province, Hangzhou Normal University, Hangzhou, China. [3]School of Computer Science and Technology, Qingdao Universit, Qingdao, China.

Received: 4 January 2022 Accepted: 16 May 2022
Published online: 03 June 2022

References

1. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public Key Encryption with Keyword Search. Springer, Heidelberg
2. Byun JW, Rhee HS, Park H-A, Lee DH (2006) Off-line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. Springer, Heidelberg
3. Baek J, Safavi-Naini R, Susilo W (2008) Public Key Encryption with Keyword Search Revisited. Springer, Heidelberg
4. Rhee HS, Park JH, Susilo W, Lee DH (2010) Trapdoor security in a searchable public-key encryption scheme with a designated tester. J Syst Softw 83(5):763–771. https://doi.org/10.1016/j.jss.2009.11.726
5. BingJian W, TzungHer C, FuhGwo J (2011) Security improvement against malicious server's attack for a dpeks scheme. Int J Inf Educ Technol 1(4):350–353
6. Tang Q, Chen L (2009) Public-key Encryption with Registered Keyword Search. Springer, Heidelberg
7. Chen R, Mu Y, Yang G, Guo F, Wang X (2015) Dual-server public-key encryption with keyword search for secure cloud storage. IEEE Trans Inf Forensic Secur 11(4):789–798. https://doi.org/10.1109/TIFS.2015.2510822
8. Huang Q, Li H (2017) An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf Sci 403:1–14. https://doi.org/10.1016/j.ins.2017.03.038
9. Qin B, Chen Y, Huang Q, Liu X, Zheng D (2020) Public-key authenticated encryption with keyword search revisited: Security model and constructions. Inf Sci 516:515–528. https://doi.org/10.1016/j.ins.2019.12.063
10. Pan X, Li F (2021) Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. J Syst Archit 115:102075. https://doi.org/10.1016/j.sysarc.2021.102075
11. Cheng L, Meng F (2021) Security analysis of pan et al.'s "public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability". J Syst Archit 119:102248. https://doi.org/10.1016/j.sysarc.2021.102248
12. Li H, Huang Q, Shen J, Yang G, Susilo W (2019) Designated-server identity-based authenticated encryption with keyword search for encrypted emails. Inf Sci 481:330–343. https://doi.org/10.1016/j.ins.2019.01.004
13. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2005) Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Annual International Cryptology Conference. Springer, Berlin, Heidelberg. pp 205–222
14. Rhee HS, Susilo W, Kim H-J (2009) Secure searchable public key encryption scheme against keyword guessing attacks. IEICE Electron Express 6(5):237–243. https://doi.org/10.1587/elex.6.237
15. Fang L, Susilo W, Ge C, Wang J (2013) Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf Sci 238:221–241. https://doi.org/10.1016/j.ins.2013.03.008
16. Rhee HS, Park JH, Lee DH (2012) Generic construction of designated tester public-key encryption with keyword search. Inf Sci 205:93–109. https://doi.org/10.1016/j.ins.2012.03.020
17. Emura K, Miyaji A, Rahman MS, Omote K (2015) Generic constructions of secure-channel free searchable encryption with adaptive security. Secur Commun Netw 8(8):1547–1560. https://doi.org/10.1002/sec.1103
18. Chen R, Mu Y, Yang G, Guo F, Huang X, Wang X, Wang Y (2016) Server-aided public key encryption with keyword search. IEEE Trans Inf Forensic Secur 11(12):2833–2842. https://doi.org/10.1109/TIFS.2016.2599293

19. Zhou Y, Xu G, Wang Y, Wang X (2016) Chaotic map-based time-aware multi-keyword search scheme with designated server. Wirel Commun Mob Comput 16(13):1851–1858. https://doi.org/10.1002/wcm.2656
20. Chen Y-C (2015) Speks: Secure server-designation public key encryption with keyword search against keyword guessing attacks. Comput J 58(4):922–933
21. Xu P, Jin H, Wu Q, Wang W (2012) Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. IEEE Trans Comput 62(11):2266–2277. https://doi.org/10.1109/TC.2012.215
22. Wang C-h, Tu T-y (2014) Keyword search encryption scheme resistant against keyword-guessing attack by the untrusted server. J Shanghai Jiaotong Univ (Sci) 19(4):440–442. https://doi.org/10.1007/s12204-014-1522-6
23. He D, Ma M, Zeadally S, Kumar N, Liang K (2017) Certificateless public key authenticated encryption with keyword search for industrial internet of things. IEEE Trans Ind Inform 14(8):3618–3627. https://doi.org/10.1109/TII.2017.2771382
24. Wu L, Zhang Y, Ma M, Kumar N, He D (2019) Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical internet of things. Ann Telecommun 74(7):423–434. https://doi.org/10.1007/s12243-018-00701-7
25. Chen X (2020) Public-key authenticate encryption with keyword search revised:\probabilistic trapgen algorithm. IACR Cryptol ePrint Arch 2020:1211
26. He D, Ma M, Zeadally S, Kumar N, Liang K (2017) Certificateless public key authenticated encryption with keyword search for industrial internet of things. IEEE Trans Ind Inform 14(8):3618–3627. https://doi.org/10.1109/TII.2017.2771382
27. Wu L, Zhang Y, Ma M, Kumar N, He D (2019) Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical internet of things. Ann Telecommun 74(7):423–434. https://doi.org/10.1007/s12243-018-00701-7
28. Pakniat N, Shiraly D, Eslami Z (2020) Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial iot. J Inf Secur Appl 53:102525. https://doi.org/10.1016/j.jisa.2020.102525
29. Lu Y, Wang G, Li J (2019) Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement. Inf Sci 479:270–276. https://doi.org/10.1016/j.ins.2018.12.004
30. Noroozi M, Eslami Z (2019) Public key authenticated encryption with keyword search: revisited. IET Inf Secur 13(4):336–342. https://doi.org/10.1049/iet-ifs.2018.5315
31. Wu L, Chen B, Zeadally S, He D (2018) An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage. Soft Comput 22(23):7685–7696. https://doi.org/10.1007/s00500-018-3224-8
32. Chen X (2020) Certificateless public-key authenticate encryption with keyword search revised: Mci and mtp. IACR Cryptol ePrint Arch 2020:1230
33. Qin B, Cui H, Zheng X, Zheng D (2021) Improved security model for public-key authenticated encryption with keyword search. In: International Conference on Provable Security. Springer, Cham. pp 19–38
34. Wang P, Xiang T, Li X, Xiang H (2020) Public key encryption with conjunctive keyword search on lattice. J Inf Secur Appl 51:102433. https://doi.org/10.1016/j.jisa.2019.102433
35. Zhang X, Tang Y, Wang H, Xu C, Miao Y, Cheng H (2019) Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage. Inf Sci 494:193–207. https://doi.org/10.1016/j.ins.2019.04.051
36. Blake IF, Seroussi G, Smart NP (2005) Advances in Elliptic Curve Cryptography vol. 317. Cambridge University Press, Cambridge
37. MIRACL (2021) The MIRACL Core Cryptographic Library. https://github.com/miracl/core/. Accessed 28 Nov 2021

Publisher's Note