Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.043589

ARTICLE

Blockchain-Based Certificateless Bidirectional Authenticated Searchable Encryption Scheme in Cloud Email System

Yanzhong Sun[1], Xiaoni Du[1,*], Shufen Niu[2] and Xiaodong Yang[2]

[1]College of Mathematics and Statistics, Northwest Normal University, Lanzhou, 730070, China

[2]College of Computer Science and Engineering, Northwest Normal University, Lanzhou, 730070, China

*Corresponding Author: Xiaoni Du. Email: ymldxn@126.com

Received: 06 July 2023 Accepted: 23 October 2023 Published: 11 March 2024

ABSTRACT

Traditional email systems can only achieve one-way communication, which means only the receiver is allowed to search for emails on the email server. In this paper, we propose a blockchain-based certificateless bidirectional authenticated searchable encryption model for a cloud email system named certificateless authenticated bidirectional searchable encryption (CL-BSE) by combining the storage function of cloud server with the communication function of email server. In the new model, not only can the data receiver search for the relevant content by generating its own trapdoor, but the data owner also can retrieve the content in the same way. Meanwhile, there are dual authentication functions in our model. First, during encryption, the data owner uses the private key to authenticate their identity, ensuring that only legal owner can generate the keyword ciphertext. Second, the blockchain verifies the data owner's identity by the received ciphertext, allowing only authorized members to store their data in the server and avoiding unnecessary storage space consumption. We obtain a formal definition of CL-BSE and formulate a specific scheme from the new system model. Then the security of the scheme is analyzed based on the formalized security model. The results demonstrate that the scheme achieves multi-keyword ciphertext indistinguishability and multi-keyword trapdoor privacy against any adversary simultaneously. In addition, performance evaluation shows that the new scheme has higher computational and communication efficiency by comparing it with some existing ones.

KEYWORDS

Cloud email system; authenticated searchable encryption; blockchain-based; designated server test; multi-trapdoor privacy; multi-ciphertext indistinguishability

1 Introduction

Email systems have become an essential component of modern communication tools and revolutionized the way we conduct business, education, and personal communication, facilitating effective and efficient communication. However, the widespread usage of email has raised significant concerns regarding email security. Searchable encryption [1], as a promising security solution, has been successfully applied in many fields, including email systems. It can not only provide users with convenient search and data management methods while preserving data privacy and security but also enrich

the overall user experience while safeguarding email confidentiality. That is, searchable encryption technology has become an indispensable security protection measure within email systems.

Public key Encryption with Keyword Search (PEKS) is a form of searchable encryption within the asymmetric category proposed by Boneh et al. [2] which optimises the security and privacy of email and improves users' experience and the system performance. Since then, several PEKS schemes with varying functionality have been proposed including secure channel-free PEKS [3] and certificateless PEKS [4,5]. Although these schemes offer numerous keyword search methods suitable for encrypted email systems, there are still some security issues to be concerned with, specifically the Keyword Guessing Attack (KGA) [6]. In fact, the limited keyword space and low entropy render most PEKS schemes vulnerable to both online and offline KGA.

To defend against KGA, Huang et al. [7] introduced the Public Key Authenticated Encryption with Keyword Search (PAEKS) as a new variant of PEKS in 2017 and proved that the proposed PAEKS scheme achieved Ciphertext Indistinguishability (CI)-secure and Trapdoor Privacy (TP)-secure. Considering against chosen multi-keyword attacks and multi-keyword guessing attacks, Qin et al. [8] presented a new security model as Multi-Ciphertext Indistinguishability (MCI) in 2020 and Pan et al. formalized Multi-Trapdoor Privacy (MTP) in [9], which are the enhancement of CI-secure and TP-secure, respectively.

It is obvious that all PEKS/PAEKS systems cannot avoid the inherent burden of certificate management and key escrow issues due to their reliance on public key infrastructure cryptosystem or identity-based cryptosystem. A common approach to overcome these problems is to incorporate the PEKS/PAEKS system in certificateless public key cryptography (CL-PKC) [10]. As a result, Peng et al. [5] proposed the first certificateless PEKS scheme and He et al. [11] developed the first certificateless PAEKS scheme. However, the CL-PKC scheme still suffers from two types of attackers. The distributed nature of blockchain makes it impossible to tamper the data stored on the chain, which solves the trust problem and guarantees data security. Therefore, in CL-PKC, in order to avoid forgery attacks launched by attackers using public parameters, part of the user's private key is created by a blockchain smart contract [12].

All of the above improvements to PEKS/PAEKS including the enhancement of security and the introduction of certificateless cryptosystem have significantly optimized their application for protecting the data security and privacy, enhancing user experience and improving system performance. Moreover, Zhang et al. [13] highlighted a crucial aspect of encrypted email systems: users must not only search for encrypted emails received from others but also retrieve encrypted emails sent to others, and they developed a new cryptographic approach named Public-key Encryption with Bidirectional Keyword Search (PEBKS). Inspired by the above ideas, it is imperative to develop a certificateless authenticated bidirectional searchable encryption scheme with a designated server test that can achieve both MCI and MTP security.

1.1 Our Contributions

The following is a list of the main contributions of this paper:

- Considering the actual application scenario of cloud email system that the data owner also needs to retrieve emails with target keywords. We apply the bidirectional searchable functionality to CL-PAEKS cryptosystem by introducing a trapdoor generation algorithm for the data owner, put forward a cryptographic concept named CL-BSE. This allows the data owner not only to encrypt and send an email to the cloud email server, but also to generate its own trapdoor for specified keyword and retrieve the corresponding email.

- On the one side, the scheme in this paper achieves bidirectional searchable functionality, on the other side, it establishes dual authentication functions. In the process of generating the ciphertext with the keyword, the data owner not only uses the public key of the data receiver, but also uses his own private key. Similarly, the data receiver uses both his own private key and the public key of the data owner to generate the corresponding trapdoor, which authenticates the identity of the data owner. Meanwhile, the blockchain can also verify the legitimacy of the ciphertexts, which effectively saves the storage space. Furthermore, the scheme satisfies designated server test which makes secure channel free.
- We formalize the definition of the new cryptographic concept CL-BSE, then give a concrete construction of the scheme under the bilinear pairing. Meanwhile, we formally define the security model of CL-BSE scheme and show that in the random oracle model, it is able to achieve both MCI and MTP security levels against inside KGA under the CBDH hardness assumption. Through the experimental comparison, our scheme has more advantages and higher efficiency in computation and communication costs.

1.2 Organization

The rest of this paper is arranged as follows. The next section presents some basic symbols and notations, including bilinear pairing and hardness assumption. Section 3 illustrates the framework of our scheme including the system model in 3.1, the formalized definition in 3.2 and the formalized security model in 3.3. Section 4 is the concrete construction of our CL-BSE scheme and Section 5 guarantees the security of the new scheme. In Section 6, we analyze the performance by comparing it with existing works. Eventually, we draw a conclusion of the paper in Section 7.

1.3 Related Works

Boneh et al. [2] presented the first PEKS scheme in 2014, which effectively solved the distribution and management of the secret-key in the symmetric searchable encryption (SSE) cryptosystem [14–16]. However, Baek et al. [3] pointed out that the trapdoor transmission channel must be secure in [2], and then proposed a secure channel free PEKS (SCF-PEKS) scheme by giving the server a public/private key pair so that only the designated server could execute the test algorithm. Rhee et al. in [17] improved the trapdoor security of [3] and then constructed a new SCF-PEKS scheme called dPEKS under the new security model. Nevertheless, Byun et al. [6] claimed that both PEKS and SCF-PEKS schemes were vulnerable to (offline) KGA and a variety of improved PEKS and SCF-PEKS schemes [18–20] were proposed to overcome the series security threats in the years including [17]. Unfortunately, it turns out that none of them can really resist the offline KGA [21]. What's worse is that Yau et al. [22] pointed out that these existing PEKS schemes suffered from another generic attack called online KGA or inside KGA in 2013.

In order to defend against both online KGA and offline KGA, Huang et al. [7] introduced a new primitive of PAEKS and proposed the first PAEKS scheme in 2017. The essence of PAEKS is to insert the data owner's public/private key pair into PEKS so that it can authenticate the keyword while encrypting it. Meanwhile, they defined the security as TP and CI for trapdoor and ciphertext, respectively. After that, Noroozi et al. [23] pointed out some weaknesses of the previous security in terms of multi-user settings. In 2020, Qin et al. [8] found that the CI & TP security in [7] does not protect the information whether two different files extract identical keywords or the same file contains how many identical keywords so they improved the security model as MCI, and they proposed a PAEKS scheme satisfying MCI instead of MTP. In 2021, Pan et al. claimed that their new scheme in [9] achieved both MCI and MTP secure until Cheng et al. [24] presented an effective attack method

on MTP. In addition to the security efforts, contributions to the functionality of PEKS/PAEKS have also been made. Fuhr et al. [25] and Hofheinz et al. [26] inserted the ciphertext decryptable function on PEKS schemes in different types of models. Zhang et al. [13] observed that in practical application, a data owner also needed to retrieve encrypted files containing specified keywords, then proposed a cryptographic system Public-key Encryption with Bidirectional Keyword Search (PEBKS) and constructed a concrete scheme.

All of the above schemes, including PEKS, PAEKS and PEBKS are identity-based cryptosystem with key escrow and certificate management issues. Peng et al. [5] constructed the first PEKS under CL-PKC named CLPEKS. In 2018, Ma et al. [4] proposed an improved CLPEKS scheme and it was improved again in the literature [27], which was been pointed out cannot achieve both MCI and MTP secure and was subsequently improved by Yang et al. [28]. But, the cryptanalysis in [29] demonstrates that these CLPEKS frameworks also suffer from the security vulnerability caused by the keyword guessing attack and in order to remedy these security weakness and provide resistance against both inside and outside keyword guessing attacks, they propose a new CLEKS scheme by embedding the owner's private key into the calculation of keyword ciphertexts, which actually is CL-PAEKS. Later, combining [4], He et al. [11] proposed a CL-PAEKS scheme, and Shiraly et al. [30] constructed a pairing-free CL-PAEKS. However, their security and functionality still need to be improved and promoted.

2 Preliminaries

2.1 Notations

The symbols and notations used in this paper are presented in the Table 1.

Table 1: The symbols and notations

Notation	Definition
1^k	System security parameter
$\mathbb{G}_1, \mathbb{G}_2$	Two cyclic groups
q	A large prime number
P_1, P_2, Q	Three distinct generators of $\mathbb{G}_1$
$H_i (1 \leq i \leq 6)$	Cryptographic hash functions
(PK_{DO}, SK_{DO})	Public/private key pair of DO
(PK_{DR}, SK_{DR})	Public/private key pair of DR
(PK_{CS}, SK_{CS})	Public/private key pair of CS
$s \leftarrow S$	Sample s from S uniformly at random
$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$	A bilinear pairing
$\text{negl}(\cdot)$	A negligible function

2.2 Bilinear Pairing

Bilinear pairing [31] is an important tool in the construction of many pairing based cryptographic schemes, including our CL-BSE scheme, and we usually construct it using the Weil pairing and the Tate pairing [31–33].

Definition 2.1 (Bilinear Pairing). Let $\mathbb{G}_1$ be an additive cyclic group of large prime order q and $\mathbb{G}_2$ a multiplicative cyclic group of the same order. A *bilinear pairing* $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a mapping which satisfies the following properties:

Bilinearity: For any $P, Q \in \mathbb{G}_1$ and any $a, b \in \mathbb{Z}_q^*$, $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$;

Non-Degeneracy: There exists a $P \in \mathbb{G}_1$ such that $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$, (where $1_{\mathbb{G}_2}$ denotes the identity in $\mathbb{G}_2$). Observe that since $\mathbb{G}_1$ and $\mathbb{G}_2$ are groups of prime order, so for any generator $P \in \mathbb{G}_1$, this statement implies that $\hat{e}(P, P) \in \mathbb{G}_2$ is a generator of $\mathbb{G}_2$;

Computability: For any $P, Q \in \mathbb{G}_1$, there is an efficient algorithm to compute $\hat{e}(P, Q)$.

2.3 Hardness Assumption

Definition 2.2 (The CBDH Assumption [31]). The Computational Bilinear Diffie-Hellman (CBDH) problem states that given $(P, aP, bP, cP) \in \mathbb{G}_1^4$, a bilinear pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, to compute $\hat{e}(P, P)^{abc}$. The CBDH assumption says that, for any PPT algorithm $\mathscr{B}$, it is hard to compute $\hat{e}(P, P)^{abc}$, given a random instance of the CBDH problem $(P, aP, bP, cP) \in \mathbb{G}_1^4$. That is,

$$\mathrm{Adv}_{\mathscr{B}}^{CBDH}(1^k) := \Pr[Z = \hat{e}(P, P)^{abc} | Z \leftarrow \mathscr{B}(P, aP, bP, cP)] \leq \mathrm{negl}(1^k), \tag{1}$$

where the probability is taken over the random choices of $P \in \mathbb{G}_1$, $a, b, c \in \mathbb{Z}_q^*$ and the random coins tossed by $\mathscr{B}$.

3 The Framework

There are three principal works in this section. First, we illustrate the system model of our protocol in this paper based on [13], and then formalise the definition of our CL-BSE scheme. Finally, we define the security model based on [8,9].

3.1 System Model

As shown in Fig. 1, the system model of our protocol includes the following five parties: cloud email server (CS), smart contract-based key generation center (SC-KGC), blockchain (BC), data owner (DO) and data receiver (DR). They are interacting as follows:

- *SC-KGC:* Deployed on the blockchain, the smart contract key generation center is a combination of a smart contract and a conventional key generation center. It is responsible for producing and storing the public parameters on the blockchain, generating and distributing partial private keys and the master key to the corresponding parties.
- *Blockchain:* To avoid the system public parameters being tampered with, blockchain stores and then transmits them to all clients. The blockchain is also responsible for verifying the validity of the ciphertext, and then transferring the verified ciphertext to the cloud server.
- *Cloud Email Server:* The cloud email server plays an "honest but curious" role in the system model, i.e., it stores the real data, retrieves keyword sets by rules and returns the corresponding results correctly. Meanwhile, it may launch keyword guessing attacks on a set of received search tokens. Furthermore, it also performs the test algorithm and then sends the search results to the data receiver.
- *Data Owner:* The data owner is a client who wants to store the encrypted data files with keyword indexes in the email server while sending it to the data receiver through the cloud email server, so that he/she can retrieve it by generating a trapdoor using his/her own private key.

- *Data Receiver:* The data receiver is one who receives emails from the cloud email server by sending a trapdoor for the keywords he/she interested in using his/her own private key.

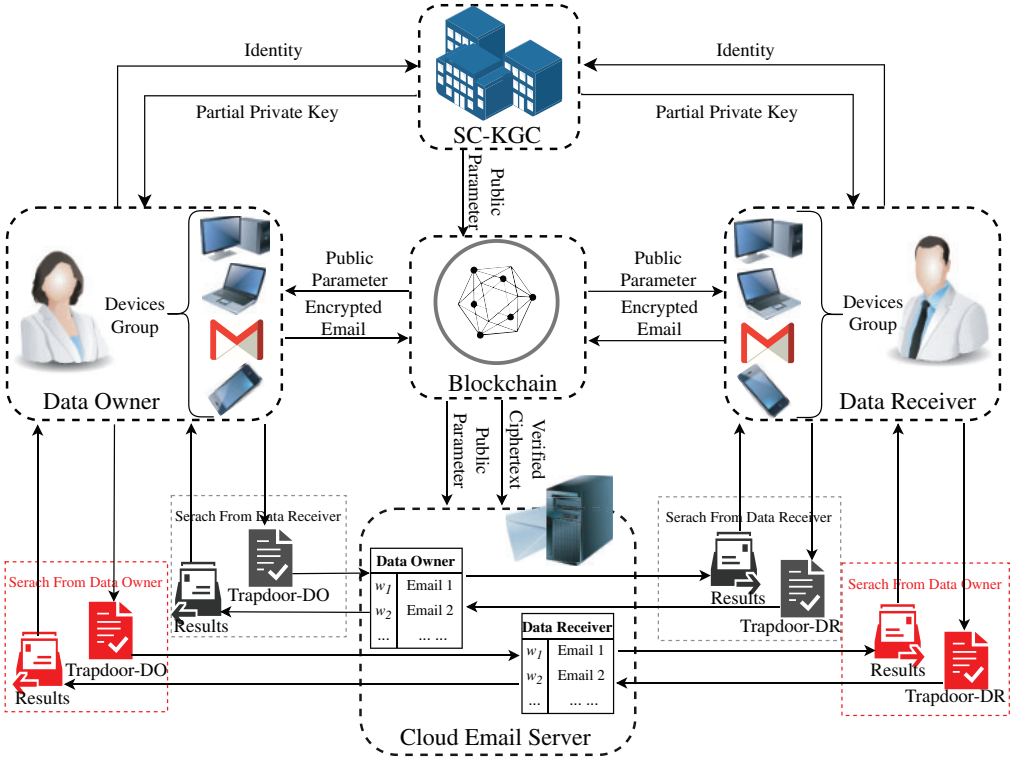

Figure 1: System model

3.2 The Definition of CL-BSE

We have formalized the architecture of our certificateless bidirectional authenticated searchable encryption (CL-BSE) scheme for the cloud email system.

Definition 3.1. The CL-BSE scheme consists essentially of nine PPT algorithms: Setup, Extract-PPK, Set-secret-value, Set-private-key, Set-public-key, CL-BSE, Trapdoor-DR, Trapdoor-DO and Test. They are described below:

- Setup (1^k): This algorithm is executed by SC-KGC. Given a security parameter 1^k, the algorithm generates the global public parameters *Params* and the master key P_{mas}.
- Extract-PPK (*Params*): This algorithm is also performed by SC-KGC. It takes as input the global public parameters *Params*, the master key P_{mas} and the client identity $ID_U (U \in \{CS, DO, DR\})$, generates and outputs each client's partial private key PPK_U.
- Set-secret-value ($Params, ID_U$): This algorithm is run by each client. Input *Params* and the client identity $ID_U (U \in \{CS, DO, DR\})$, it generates the secret value SV_U for each participant.
- Set-private-key ($Params, SV_U, PPK_U$): Each client runs this algorithm on its own. Entering *Params*, secret value SV_U and partial private key PPK_U, it outputs the private key SK_U for each client.

- Set-public-key ($Params, SV_U$): This algorithm is executed by each client. It takes as input $Params$ and the secret value SV_U, and outputs the public key PK_U for itself.
- CL-BSE ($PK_{DR}, PK_{CS}, SK_{DO}, w$): This is the keyword ciphertext generation algorithm and is performed by the data owner. It takes as input PK_{DR}, PK_{CS}, SK_{DO} and keyword w with respect to the encrypted files, outputs a ciphertext C_w.
- Trapdoor-DR ($PK_{DO}, PK_{CS}, SK_{DR}, w'$): This algorithm generates trapdoor for data receiver. When searching the encrypted files containing the keyword w' from the cloud email server, it takes as input PK_{DO}, PK_{CS}, SK_{DR} and the keyword w', and outputs T_{DR}.
- Trapdoor-DO ($PK_{DR}, PK_{CS}, SK_{DO}, w'$): This algorithm generates trapdoor for data owner. When the client wants to retrieve the encrypted files containing the keyword w' from the cloud email server, it takes as input PK_{DR}, PK_{CS}, SK_{DO} and the keyword w', outputs T_{DO}.
- Test ($C_w, SK_{CS}, T_{w'} = T_{DO}(T_{DR})$): The test algorithm is executed by the cloud email server. It takes C_w and $T_{w'}$ as input and returns "1" if $w = w'$ and "0" otherwise.

Correctness. The correctness of our CL-BSE scheme is defined as follows. For any legally registered clients $ID_U(U \in \{DO, DR, CS\})$ with public/private key pairs (PK_U, SK_U). Let $C_w \leftarrow CL - BSE(PK_{DR}, PK_{CS}, SK_{DO}, w)$ be the ciphertext of w, $T_{DO} \leftarrow Trapdoor - DO(PK_{DR}, PK_{CS}, SK_{DO}, w')$ and $T_{DR} \leftarrow Trapdoor - DR(PK_{DO}, PK_{CS}, SK_{DR}, w')$ be the trapdoors of w' generated by DO and DR, respectively. Correctness implies

$$\Pr[Test(Params, T_{DO}(w'), C_w) = 1] = \Pr[Test(Params, T_{DR}(w'), C_w) = 1] = 1. \quad (2)$$

3.3 Security Model

As with other certificateless cryptosystems [5,10,28,34,35], the CL-BSE scheme considers two types of adversary with different privileges: Type-I adversary $\mathscr{A}_1$ and Type-II adversary $\mathscr{A}_2$. Specifically,

- **Type-I adversary** $\mathscr{A}_1$. It plays the part of the malicious user who is available to perform queries including extract partial private key, request public key, extract secret value and replace public key, but does not have access to the master private key P_{mas}.
- **Type-II adversary** $\mathscr{A}_2$. It models an honest-but-curious SC-KGC that has access to the master private key P_{mas}, but not allowed to replace public key query.

Now, the above queries are listed as follows, which are actually interactions between an adversary $\mathscr{A}_1(\mathscr{A}_2)$ and a challenger $\mathscr{C}$.

– *Extract-PPK query*. When $\mathscr{A}_1(\mathscr{A}_2)$ queries partial private key for identity ID_i, $\mathscr{C}$ executes Extract-PPK algorithm and returns partial private key PPK_i.

– *Extract-secret-value query*. When $\mathscr{A}_1(\mathscr{A}_2)$ queries secret value for identity ID_i, $\mathscr{C}$ executes Set-secret-value algorithm and returns a secret value SV_i.

– *Request-public-key query*. When $\mathscr{A}_1(\mathscr{A}_2)$ queries public key for identity ID_i, $\mathscr{C}$ executes Set-public-key algorithm and returns public key PK_i.

– *Replace-public-key query*. $\mathscr{A}_1$ is permitted to ask $\mathscr{C}$ to replace the public PK_i with a new one PK_i' for any user ID_i.

– *Ciphertext query*. When $\mathscr{A}_1(\mathscr{A}_2)$ queries the ciphertext of the keyword w, the challenger $\mathscr{C}$ returns the matching ciphertext C_w.

– *Data receiver trapdoor query*. When $\mathscr{A}_1(\mathscr{A}_2)$ queries the data receiver trapdoor of keyword w', $\mathscr{C}$ returns the matching trapdoor $T_{DR} = T_{w'}$.
– *Data owner trapdoor query*. When $\mathscr{A}_1(\mathscr{A}_2)$ queries the data owner trapdoor of keyword w', $\mathscr{C}$ returns the matching trapdoor $T_{DO} = T_{w'}$.

In order to capture chosen multi-keyword attacks and multi-keyword guessing attacks, the security model of our CL-BSE scheme are defined as MCI [8] and MTP [9], which are the enhancement of CI-secure and TP-secure [7], respectively. Their formal definitions are described by the following games, which are interactions between an challenger $\mathscr{C}$ and an adversary $\mathscr{A}_1$ or $\mathscr{A}_2$.

Game 1: The MCI Security against Adversary $\mathscr{A}_1$.

- *Setup*. Given security parameter 1^k, $\mathscr{C}$ generates public parameter *Params* and system master key P_{mas} by running Setup algorithm. It then responds only to $\mathscr{A}_1$ *Params* and keeps master key P_{mas} secret.
- *Phase 1*. $\mathscr{A}_1$ can adaptively perform a series of polynomial times queries, including *Extract-PPK* query, *Extract-secret-value* query, *Request-public-key* query, *Replace-public-key* query, *Ciphertext* query, *Data receiver trapdoor* query and *Data owner trapdoor* query.
- *Challenge*. After $\mathscr{A}_1$ finishes the queries in *Phase 1*, it selects two challenge keyword sets $\mathbf{w}_0^* = \{w_{0,1}, w_{0,2}, \ldots, w_{0,n}\}$, $\mathbf{w}_1^* = \{w_{1,1}, w_{1,2} \ldots, w_{1,n}\}$ which are not queried in *Phase 1* and sends them to $\mathscr{C}$. After that, it first chooses a random bit $b \in \{0, 1\}$, then computes the searchable ciphertext $C_{\mathbf{w}_b^*}$ with respect to $\mathbf{w}_b^*$. Finally, it returns $C_{\mathbf{w}_b^*}$ as a challenge ciphertext.
- *Phase 2*. As in *Phase 1*, $\mathscr{A}_1$ can continue to make series queries for polynomial times and the restriction here is that cannot make ciphertext query and two trapdoor queries on $\mathbf{w}_0^*$ and $\mathbf{w}_1^*$.
- *Guess*. Finally, $\mathscr{A}_1$ outputs its guess $b' \in \{0, 1\}$ on b, and wins this game if $b' = b$.

The following probability equation defines $\mathscr{A}_1$'s advantage in the game,

$$\mathrm{Adv}_{\mathscr{A}_1}^{MCI}(1^k) = \left| \Pr[b' = b] - \frac{1}{2} \right|. \tag{3}$$

Game 2: The MCI Security against Adversary $\mathscr{A}_2$.

- *Setup*. Given security parameter 1^k, $\mathscr{C}$ runs Setup algorithm generates public parameter *Params* and system master key P_{mas}, then sends both *Params* and P_{mas} to $\mathscr{A}_2$.
- *Phase 1*. As in *Game 1*, $\mathscr{A}_2$ can make *Extract-PPK* query, *Extract-secret-value* query, *Request-public-key* query, *Ciphertext* query, *Data receiver trapdoor* query and *Data owner trapdoor* query, but not available for *Replace-public-key* query.
- *Challenge*. When $\mathscr{A}_2$ completed the *Phase 1*, it selects $\mathbf{w}_0^*$, $\mathbf{w}_1^*$ as challenge keywords sets like the steps in *Game 1*, and sends them to $\mathscr{C}$. Then $\mathscr{C}$ chooses a random bit $b \in \{0, 1\}$ and generates the ciphertext $C_{\mathbf{w}_b^*}$ with respect to $\mathbf{w}_b^*$. Finally, it returns $C_{\mathbf{w}_b^*}$ to $\mathscr{A}_2$ as a challenge ciphertext.
- *Phase 2*. This phase is the same as *Game 1*, $\mathscr{A}_2$ is not allowed to make ciphertext query and two trapdoor queries on $\mathbf{w}_0^*$ and $\mathbf{w}_1^*$.
- *Guess*. Finally, $\mathscr{A}_2$ outputs its guess $b' \in \{0, 1\}$ on b, and wins if $b' = b$.

The advantage of $\mathscr{A}_2$ in *Game 2* is defined by the following probability equation:

$$\mathrm{Adv}_{\mathscr{A}_2}^{MCI}(1^k) = \left| \Pr[b' = b] - \frac{1}{2} \right|. \tag{4}$$

Definition 3.2 (MCI security). The CL-BSE scheme is said MCI security if for any PPT adversary $\mathscr{A}$, its advantages $\mathrm{Adv}_{\mathscr{A}_1}^{MCI}(1^k)$ and $\mathrm{Adv}_{\mathscr{A}_2}^{MCI}(1^k)$ against the challenger $\mathscr{C}$ in *Game 1* and *Game 2* are negligible.

Game 3: The MTP Security against Adversary $\mathscr{A}_1$.

- *Setup*. The setup algorithm is the same as *Game 1*, $\mathscr{C}$ also only sends the public parameter *Params* to $\mathscr{A}_1$ eventually.
- *Phase 1*. Same as process *Phase 1* in *Game 1*.
- *Challenge*. When $\mathscr{A}_1$ has finished the *Phase 1*, do the same as that in *Game 1* to obtain two challenge keywords sets $\mathbf{w}_0^*$, $\mathbf{w}_1^*$ and send them to $\mathscr{C}$. After that, $\mathscr{C}$ chooses a random bit $b \in \{0,1\}$ first, then computes the trapdoor $T_{\mathbf{w}_b^*}$ with respect to $\mathbf{w}_b^*$. Finally, it returns $T_{\mathbf{w}_b^*}$ to $\mathscr{A}_1$ as a challenge trapdoor.
- *Phase 2*. Same as process *Phase 2* in *Game 1*.
- *Guess*. Finally, $\mathscr{A}_2$ outputs its guess $b' \in \{0,1\}$ on b, and wins this game if $b' = b$.

The advantage of $\mathscr{A}_1$ in *Game 3* is defined by

$$\mathrm{Adv}_{\mathscr{A}_1}^{MTP}(1^k) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \tag{5}$$

Game 4: The MTP Security against Adversary $\mathscr{A}_2$.

- *Setup*. The setup algorithm is the same as *Game 2*, the challenger $\mathscr{C}$ sends both the public parameter *Params* and the master key P_{mas} to $\mathscr{A}_2$ eventually.
- *Phase 1*. Same as process *Phase 1* in *Game 2*.
- *Challenge*. After $\mathscr{A}_2$ has finished the *Phase 1*, do the same as that in *Game 3* to obtain two challenge keyword sets $\mathbf{w}_0^*$, $\mathbf{w}_1^*$ and send them to $\mathscr{C}$. Then $\mathscr{C}$ chooses a random bit $b \in \{0,1\}$ and generates the trapdoor $T_{\mathbf{w}_b^*}$ with respect to $\mathbf{w}_b^*$. Finally, it returns $T_{\mathbf{w}_b^*}$ to $\mathscr{A}_2$ as a challenge trapdoor.
- *Phase 2*. Same as process *Phase 2* in *Game 2*.
- *Guess*. Finally, $\mathscr{A}_2$ outputs its guess $b' \in \{0,1\}$ on b, and it wins this game if $b' = b$.

The advantage of $\mathscr{A}_2$ in *Game 4* is defined by

$$\mathrm{Adv}_{\mathscr{A}_2}^{MTP}(1^k) = \left|\Pr[b' = b] - \frac{1}{2}\right|. \tag{6}$$

Definition 3.3 (MTP security). The CL-BSE scheme is said MTP security for both data receiver trapdoor and data owner trapdoor, if for any PPT adversary $\mathscr{A}$, its advantages $\mathrm{Adv}_{\mathscr{A}_1}^{MTP}(1^k)$ and $\mathrm{Adv}_{\mathscr{A}_2}^{MTP}(1^k)$ against the challenger $\mathscr{C}$ in *Game 3* and *Game 4* are negligible.

4 The Proposed CL-BSE Scheme

In this section, we give a concrete construction as the formal definition of the CL-BSE scheme in Section 3.2 with a designated server, it consists of nine PPT algorithms in five phases: System initialization, Key generation, Keyword encryption, Trapdoor generation and Test.

Phase A. System Initialization.

- Setup (1^k): Given the security parameter 1^k, SC-KGC performs the following steps:

(1) Select a cyclic additive group $\mathbb{G}_1$, a cyclic multiplicative group $\mathbb{G}_2$ with a large prime order $q, (q > 2^k)$ and three generators P_1, P_2 and $Q \in \mathbb{G}_1$, generate a bilinear pair $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$;

(2) Pick a random number $s \leftarrow \mathbb{Z}_q^*$, put it as the system master key and store it secretly, then compute $P_{pub} = sP_1$;

(3) Define six different cryptographic hash functions $H_i (1 \leq i \leq 6)$ as: $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_3 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_4 : \mathbb{G}_1 \to \{0,1\}^{len}$, where *len* is the fixed length output, $H_5 : \mathbb{G}_2 \to \mathbb{Z}_q^*$, $H_6 : \{0,1\}^{\log w + len} \to \mathbb{Z}_q^*$, and log$w$ denotes the length of w;

(4) Broadcast the public parameters $params = \{1^k, \hat{e}, \mathbb{G}_1, \mathbb{G}_2, q, P_1, P_2, Q, P_{pub}, H_i (1 \leq i \leq 6)\}$ on the blockchain.

Phase B. Key Generation.

- Extract-PPK (*Params*): SC-KGC takes as input the public parameters *Params*, the master key P_{mas} and the identity $ID_U (U \in \{CS, DO, DR\})$, then generates partial private keys for all clients as follows:
  (1) For *CS*, SC-KGC selects $r_{CS} \leftarrow \mathbb{Z}_q^*$ randomly, computes $R_{CS} = r_{CS} P_2$, $\alpha_{CS} = H_2(ID_{CS}, R_{CS})$ and $d_{CS} = r_{CS} + \alpha_{CS} s (\text{mod} \sim q)$, and outputs $PPK_{CS} = (R_{CS}, d_{CS})$ as its partial private key;
  (2) For $U \in \{DO, DR\}$, SC-KGC computes their partial private key as $PPK_U = D_U$, where $D_U = sQ_U$ and $Q_U = H_1(ID_U)$.
- Set-secret-value (*Params*, ID_U): The client $U \in \{DO, DR\}$ selects $x_U, y_U \leftarrow \mathbb{Z}_q^*$ randomly and sets $SV_U = (x_U, y_U)$. *CS* selects a single random number $x_{CS} \leftarrow \mathbb{Z}_q^*$ and sets $SV_{CS} = x_{CS}$.
- Set-private-key (*Params*, SV_U, PPK_U): The client $U \in \{DO, DR\}$ and *CS* set their own private keys as $SK_U = (x_U, y_U, D_U)$ and $SK_{CS} = (x_{CS}, d_{CS})$, respectively.
- Set-public-key (*Params*, SV_U): The client $U \in \{DO, DR\}$ computes $P_U = x_U P_1$, $Y_U = y_U Q$, and assigns its public keys as $PK_U = (P_U, Y_U)$, while *CS* assigns its public key as $PK_{CS} = (P_{CS}, R_{CS})$, where $P_{CS} = x_{CS} P_2$.

Phase C. Keywords Encryption.

- CL-BSE (PK_{DR},PK_{CS},SK_{DO},w): When *DO* wants to encrypt the keyword w extracted from the encrypted emails, he/she enters the relevant parameters *params*, SK_{DO}, ID_{DR}, PK_{DR}, ID_{CS}, PK_{CS} and performs the following steps:
  (1) Compute $\alpha_{CS} = H_2(ID_{CS}, R_{CS})$, $\beta_{CS} = H_3(ID_{CS}, P_{pub}, P_{CS}, R_{CS})$;
  (2) Compute $k_1 = H_4(y_{DO} Y_{DR})$, $k_2 = H_5(\hat{e}(D_{DO}, Q_{DR}))$;
  (3) Select $r \leftarrow \mathbb{Z}_q^*$ randomly, then compute

$$C_1 = r \cdot P_2, \tag{7}$$

$$C_2 = \hat{e}(P_{DR}, \beta_{CS} P_{CS} + R_{CS} + \alpha_{CS} P_{pub})^{r x_{DO} H_6(w \| k_1)}, \tag{8}$$

$$C_3 = r \cdot k_2 \cdot P_1; \tag{9}$$

  (4) Compute $V = D_{DO} + (r \cdot k_2 + x_{DO}) R_{CS}$;
  (5) Upload the ciphertext $C_w = (C_1, C_2, C_3, V)$ on the blockchain.

Upon receiving $C_w = (C_1, C_2, C_3, V)$ from the data owner, the blockchain verifies the owner's legitimacy by the equation

$$\hat{e}(V, P_1) \stackrel{?}{=} \hat{e}(Q_{DO}, P_{pub})\hat{e}(C_3 + P_{DO}, R_{CS}). \tag{10}$$

If and only if the owner is a legal member of the system, the blockchain then stores the verified ciphertexts to the cloud server, which can effectively save storage space.

Phase D. Trapdoor Generation.

Both DR and DO can generate their own trapdoor in the following ways:

- Trapdoor-DR (PK_{DO},PK_{CS},SK_{DR},w'): Input parameters $Params$, SK_{DR}, PK_{DO} and PK_{CS}, when DR searches for the files containing the keyword w', it performs the following operations:
  (1) Recall α_{CS}, β_{CS}, and $k_1 = H_4(y_{DR} Y_{DO})$, $k_2 = H_5(\hat{e}(D_{DR}, Q_{DO}))$;
  (2) Select a random number $t_{DR} \leftarrow \mathbb{Z}_q^*$ and compute

$$T_1 = t_{DR}(\beta_{CS} P_{CS} + R_{CS} + \alpha_{CS} P_{pub}), \tag{11}$$

$$T_2 = t_{DR} k_2 P_1 + x_{DR} H_6(w' \| k_1) P_{DO}; \tag{12}$$

  (3) Output $T_{DR} = (T_1, T_2)$.
- Trapdoor-DO ($PK_{DR}, PK_{CS}, SK_{DO}, w'$). Different from other CL-PAEKS models, the bidirectional keyword search functionality in this paper is achieved by introducing DO's trapdoor generation algorithm. In fact, similar to DR's trapdoor generation process above, when DO retrieves the data from CS, it does not need to generate any additional variables, but simply uses its own private key SK_{DO} and PK_{DR}, PK_{CS} to generate the trapdoor $T_{DO} = (T_1, T_2)$. That is,

$$T_1 = t_{DO}(\beta_{CS} P_{CS} + R_{CS} + \alpha_{CS} P_{pub}), \tag{13}$$

$$T_2 = t_{DO} k_2 P_1 + x_{DO} H_6(w' \| k_1) P_{DR}. \tag{14}$$

Phase E. Test Process.

- Test ($C_w, SK_{CS}, T_{w'} = T_{DO}/T_{DR}$): Take C_w, SK_{CS}, $T_{w'} = T_{DO}$ or T_{DR} as input, CS verifies whether

$$\hat{e}(T_2, (\beta_{CS} x_{CS} + d_{CS}) C_1) \stackrel{?}{=} \hat{e}(T_1, C_3) \cdot C_2 \tag{15}$$

  holds. Output "1" if it holds and "0" otherwise.

Actually, from the verification equation it can be seen that since x_{CS} and d_{CS} are private keys secretly held by the cloud email server, then only the server holding the private key can verify the equation above, i.e., our scheme is a bidirectional searchable encryption scheme with secure channel free for designated server verification.

Correctness.

(1) $\hat{e}(V, P_1) \stackrel{?}{=} \hat{e}(Q_{DO}, P_{pub})\hat{e}(C_3 + P_{DO}, R_{CS})$.

$$\begin{aligned}
\hat{e}(V, P_1) &= \hat{e}(D_{DO} + (rk_2 + x_{DO}) R_{CS}, P_1) \\
&= \hat{e}(D_{DO}, P_1)\hat{e}((rk_2 + x_{DO}) R_{CS}, P_1) \\
&= \hat{e}(D_{DO}, P_1)\hat{e}((rk_2 + x_{DO}) P_1, R_{CS}) \\
&= \hat{e}(Q_{DO}, P_{pub})\hat{e}(C_3 + P_{DO}, R_{CS}).
\end{aligned}$$

(2) $\hat{e}(T_2, (\beta_{CS}x_{CS} + d_{CS})C_1) \stackrel{?}{=} \hat{e}(T_1, C_3) \cdot C_2$.

$$\begin{aligned}
&\hat{e}(T_2, (\beta_{CS}x_{CS} + d_{CS})C_1)\\
&= \hat{e}(t_{DO}k_2P_1 + x_{DO}H_6(w\|k_1)P_{DR}, r(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)\\
&= \hat{e}(t_{DO}k_2P_1, r(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)\hat{e}(x_{DO}H_6(w\|k_1)P_{DR}, r(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)\\
&= \hat{e}(t_{DO}k_2P_1, r(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)\hat{e}(x_{DO}H_6(w\|k_1)x_{DR}P_1, r(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)\\
&= \hat{e}(P_1, P_2)^{t_{DO}k_2r(\beta_{CS}x_{CS}+r_{CS}+s\alpha_{CS})}\hat{e}(P_1, P_2)^{rx_{DO}x_{DR}H_6(w\|k_1)(\beta_{CS}x_{CS}+r_{CS}+s\alpha_{CS})},
\end{aligned}$$

and

$$\begin{aligned}
&\hat{e}(T_1, C_3) \cdot C_2\\
&= \hat{e}(t_{DO}(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2, rk_2P_1)\hat{e}(P_{DR}, (\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)^{rx_{DO}H_6(w\|k_1)}\\
&= \hat{e}(t_{DO}(\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2, rk_2P_1)\hat{e}(x_{DR}P_1, (\beta_{CS}x_{CS} + r_{CS} + s\alpha_{CS})P_2)^{rx_{DO}H_6(w\|k_1)}\\
&= \hat{e}(P_1, P_2)^{rt_{DO}k_2(\beta_{CS}x_{CS}+r_{CS}+s\alpha_{CS})}\hat{e}(P_1, P_2)^{rx_{DO}x_{DR}H_6(w\|k_1)(\beta_{CS}x_{CS}+r_{CS}+s\alpha_{CS})}.
\end{aligned}$$

The verification process described above demonstrates the correctness of the data owner's trapdoor and ciphertext test, the verification with respect to the data receiver's trapdoor is similar and we omit it.

5 Security Analysis

Based on the formal definition of security models in Section 3.3 and the CBDH hardness assumption in Section 2.3, we give the security proof of our scheme in this section.

Theorem 5.1 (MCI security). In the random oracle model, our CL-BSE scheme achieves semantically MCI security against outside chosen multi-keyword attacks under the CBDH hardness assumption.

The proof of Theorem 5.1 can be achieved by the following two lemmas.

Lemma 5.1. In the random oracle model, for any PPT adversary $\mathscr{A}_1$, there is an algorithm $\mathscr{B}$ that can break the CBDH assumption with advantage

$$\varepsilon' \geq \frac{2\varepsilon}{q_{H_1}}\left(1 - \frac{1}{q_{H_1}}\right)^{q_{E_1}+q_{E_2}+q_C+q_T} \tag{16}$$

if $\mathscr{A}_1$ wins Game 1 with advantage ε.

Proof. Give a bilinear pair $(\hat{e}, \mathbb{G}_1, \mathbb{G}_2, P_1, q)$ and an instance of CBDH assumption $(P_1, aP_1, bP_1, cP_1) \in \mathbb{G}_1^4$, algorithm $\mathscr{B}$ calculates $\hat{e}(P_1, P_1)^{abc}$ by taking $\mathscr{A}_1$ as a subroutine as follows:

- *Setup*. For a given security parameter 1^k, $\mathscr{B}$ generates public parameter $Params = \{\mathbb{G}_1, \mathbb{G}_2, q, P_1, P_2 = \alpha P_1, Q, \hat{e}, P_{pub}, H_i(1 \leq i \leq 6)\}$ and system master key $P_{mas} = s \in \mathbb{Z}_q^*$ firstly, then sets $P_{DO} = aP_1$, $P_{DR} = bP_1$, $PK_{CS} = (P_{CS}, R_{CS})$ and $SK_{CS} = (x_{CS}, d_{CS})$, chooses $ID_I(1 \leq I \leq q_{H_1})$ randomly as the challenge identity. Finally, it only responds $(Params, P_{DO}, P_{DR}, PK_{CS}, SK_{CS})$ to $\mathscr{A}_1$, and keeps P_{mas} secretly.
- *Phase 1*. $\mathscr{A}_1$ preforms a series of queries with polynomially many times adaptively, they are

– H_1-*query*. $\mathscr{B}$ maintains a list $L_{H_1} = \{\langle ID_i, \lambda_i, Q_i\rangle\}$ to respond H_1-query. For any ID_i submitted by $\mathscr{A}_1$, $\mathscr{B}$ first checks whether the ID_i already exists on L_{H_1} in a tuples $\langle ID_i, \lambda_i, Q_i\rangle$, outputs corresponding Q_i if so, otherwise selects $\lambda_i \leftarrow \mathbb{Z}_q^*$ randomly, outputs $Q_i = \lambda_i P_1$, and adds $\langle ID_i, \lambda_i, Q_i\rangle$ to L_{H_1}.
– H_2-*query*. $\mathscr{B}$ maintains a list $L_{H_2} = \{\langle ID_i, R_i, \alpha_i\rangle\}$ to respond H_2-query. For any $(ID_i, R_i) \in \{0,1\}^* \times \mathbb{G}_1$ submitted by $\mathscr{A}_1$, $\mathscr{B}$ checks whether (ID_i, R_i) already exists on L_{H_2} in a tuples $\langle ID_i, R_i, \alpha_i\rangle$ firstly, outputs corresponding α_i if so, otherwise selects and outputs $\alpha_i \leftarrow \mathbb{Z}_q^*$ randomly, and adds $\langle ID_i, R_i, \alpha_i\rangle$ to L_{H_2} at the same time.
– H_3-*query*. $\mathscr{B}$ maintains a list $L_{H_3} = \{\langle ID_i, P_i, R_i, \beta_i\rangle\}$ to respond H_3-query. For any (ID_i, P_i, R_i) submitted by $\mathscr{A}_1$, $\mathscr{B}$ first checks whether the tuples including (ID_i, P_i, R_i) already exists on L_{H_3}, outputs corresponding β_i if so, otherwise selects and outputs $\beta_i \leftarrow \mathbb{Z}_q^*$ randomly, and adds $\langle ID_i, P_i, R_i, \beta_i\rangle$ to L_{H_3} at the same time.
– H_4-*query*. $\mathscr{B}$ maintains a list $L_{H_4} = \{\langle M_i, k_{1i}\rangle\}$ to respond H_4-query. For any $M_i \in \mathbb{G}_1$ submitted by $\mathscr{A}_1$, $\mathscr{B}$ first checks whether the M_i already exists on L_{H_4} in a tuples $\langle M_i, k_{1i}\rangle$, outputs k_{1i} if so, otherwise selects $k_{1i} \leftarrow \mathbb{Z}_q^*$ randomly and outputs it, then add $\langle M_i, k_{1i}\rangle$ to L_{H_4}.
– H_5-*query*. $\mathscr{B}$ maintains a list $L_{H_5} = \{\langle N_i, k_{2i}\rangle\}$ to respond H_4-query. For any $N_i \in \mathbb{G}_2$ submitted by $\mathscr{A}_1$, $\mathscr{B}$ first checks whether the N_i already exists on L_{H_5} in a tuples $\langle N_i, k_{2i}\rangle$, outputs k_{2i} if so, otherwise selects $k_{2i} \leftarrow \mathbb{Z}_q^*$ randomly and outputs it, then adds $\langle N_i, k_{2i}\rangle$ to L_{H_5}.
– H_6-*query*. $\mathscr{B}$ maintains a list $L_{H_6} = \{\langle I_i, \eta_i\rangle\}$ to respond H_4-query, where $I_i = w_i \| k_{1i} \in \{0,1\}^*$, $k_{1i} \in L_{H_4}$. For any I_i submitted by $\mathscr{A}_1$, $\mathscr{B}$ first checks whether the I_i already exists on L_{H_6} in a tuples $\langle I_i, \eta_i\rangle$, outputs η_i if so, otherwise selects $\eta_i \leftarrow \mathbb{Z}_q^*$ randomly and outputs it, then adds $\langle I_i, \eta_i\rangle$ to L_{H_6}.
– *Extract-PPK query*. $\mathscr{B}$ maintains a list $L_{E_1} = \{\langle ID_i, Q_i, D_i\rangle\}$ to respond $\mathscr{A}_1$ for the partial private key of ID_i. It performs H_1-*query* and gets $\langle ID_i, \lambda_i, Q_i\rangle$ first and adds them to L_{H_1}. If $ID_i \neq ID_I$, then $\mathscr{B}$ computes $D_i = \lambda_i P_{pub}$ and outputs D_i, and adds $\langle ID_i, Q_i, D_i\rangle$ into L_{E_1}, otherwise aborts the query and denotes the event as E_1.
– *Request-public-key query*. $\mathscr{B}$ maintains a list $L_{E_2} = \{\langle ID_i, x_i, y_i, P_i, Y_i\rangle\}$ to respond $\mathscr{A}_1$ for the public key of ID_i. It first checks whether ID_i already exists on L_{E_2}, retrieves $PK_i = (P_i, Y_i)$ if so, otherwise selects two random numbers $x_i, y_i \leftarrow \mathbb{Z}_q^*$, computes $P_i = x_i P_1$, $Y_i = y_i P_1$, and outputs $PK_i = (P_i, Y_i)$, meanwhile, adds $\langle ID_i, x_i, y_i, P_i, Y_i\rangle$ into L_{E_2}.
– *Replace-public-key query*. When $\mathscr{B}$ receives a new public key $PK_i' = (P_i', Y_i')$ of identity ID_i queried by $\mathscr{A}_1$, it updates the original tuple $\langle ID_i, x_i, y_i, P_i, Y_i\rangle$ in L_{E_2} with $\langle ID_i, \perp, \perp, P_i', Y_i'\rangle$.
– *Extract-private-key query*. When the identity ID_i is queried by $\mathscr{A}_1$, $\mathscr{B}$ checks whether $ID_i = ID_I$ first. If $ID_i \neq ID_I$, then it performs as follows, if ID_i already exists on L_{E_1} and L_{E_2} in their tuples $\langle ID_i, Q_i, D_i\rangle$ and $\langle ID_i, x_i, y_i, P_i, Y_i\rangle$ respectively, then $\mathscr{B}$ retrieves $SK_i = (x_i, y_i, D_i)$ and responds to $\mathscr{A}_1$, otherwise performs *Extract-PPK query* and *Request-public-key query* with ID_i and retrieves corresponding $SK_i = (x_i, y_i, D_i)$. If $ID_i = ID_I$, then it aborts and denotes this event as E_2.
– *Ciphertext query*. Upon $\mathscr{B}$ receiving a query $(w', ID'_{DO}, ID'_{DR}, ID'_{CS})$ about ciphertext. If $ID'_{DR} \neq ID_I$, then it selects $r' \leftarrow \mathbb{Z}_q^*$ randomly, extracts the corresponding value from the above lists, and computes

$$C_1' = r' P_1, \tag{17}$$

$$C_2' = \hat{e}(P'_{DR}, \beta' P'_{CS} + R'_{CS} + \alpha'_{CS} P_{pub})^{r' x'_{DO} \eta'_i}, \tag{18}$$

$$C_3' = r' k_{2i}' P_1, \tag{19}$$

$$V' = D_{DO}' + (r' k_{2i}' + x_{DO}') R_{CS}'. \tag{20}$$

Otherwise $\mathscr{B}$ aborts and denotes it as E_3. Finally, it outputs $C_w' = (C_1', C_2', C_3', V')$ to $\mathscr{A}_1$.

– *Trapdoor query*. No matter the trapdoor is queried to be generated by the data receiver or the data owner. $\mathscr{B}$ performs the following steps. Upon receiving a query $(w', ID_{DO}', ID_{DR}', ID_{CS}')$ for a trapdoor, it aborts and denotes it as E_4 if and only if $ID_{DR}' = ID_I$. Otherwise $\mathscr{B}$ selects $t' \leftarrow \mathbb{Z}_q^*$ randomly, extracts the corresponding value from the above lists, and computes

$$T_1 = t'(\beta' P_{CS}' + R_{CS}' + \alpha_{CS}' P_{pub}), \tag{21}$$

$$T_2 = t' k_{2i}' P_1 + x_{DR}' \eta_i' P_{DO}'. \tag{22}$$

Finally, it outputs $T_w' = (T_1', T_2')$ to $\mathscr{A}_1$.

- *Challenge*. After $\mathscr{A}_1$ finished *Phase 1* queries, it selects $\mathbf{w}_0^*$, $\mathbf{w}_1^*$ not be queried in *Phase 1* as the challenge keywords sets and the challenge identity ID_I, then sends them to $\mathscr{B}$, $\mathscr{B}$ checks whether $ID_{DR}' = ID_I$, if not, $\mathscr{B}$ aborts and denotes it as E_5, and gives a random bit b' as a guess of b. Otherwise, it first chooses a random bit $b \in \{0,1\}$ and a random number $r^* \leftarrow \mathbb{Z}_q^*$, computes $C_1^* \in \mathbb{G}_1$, $C_2^* = \hat{e}(P_{DR}', C_3^*)^{x_{DO}' \eta_i'}$, $C_3^* = r^* k_{2i}' c P_1$ and $V^* \in \mathbb{G}_1$. Finally, it responds $\mathscr{A}_1$ with $C_{\mathbf{w}_b}^* = (C_1^*, C_2^*, C_3^*, V^*)$.
- *Phase 2*. As in *Phase 1*, $\mathscr{A}_1$ can make a series of queries for polynomial times and it can not make ciphertext query and two trapdoor queries on any keyword in $\mathbf{w}_0^*$ and $\mathbf{w}_0^*$. Denote this event as E_6.
- *Guess*. Finally, $\mathscr{A}_1$ outputs its guess $b' \in \{0,1\}$ on b, and $\mathscr{B}$ chooses $C_{\mathbf{w}_b}^* = (C_1^*, C_2^*, C_3^*, V)$ to compute $\left(C_2^*\right)^{\frac{1}{r^* k_{2i}' \eta_i'}}$. If $\mathscr{A}_1$ is correct, then no matter $b = 0$ or $b = 1$, it can compute $C_2^* = \hat{e}(P_{DR}', C_3^*)^{x_{DO}' \eta_i'}$. Furthermore, for unknown $a, b, c \in \mathbb{Z}_q^*$, we set $P_{DO}' = aP_1$, $P_{DR}' = bP_1$, and $C_3^* = r^* k_{2i}' c P_1$, $\mathscr{B}$ can compute $\hat{e}(P_1, P_1)^{abc}$ as

$$\begin{aligned}
\left(C_2^*\right)^{\frac{1}{r^* k_{2i}' \eta_i'}} &= \left(\hat{e}(P_{DR}', C_3^*)^{x_{DO}' \eta_i'}\right)^{\frac{1}{r^* k_{2i}' \eta_i'}} \\
&= \left(\hat{e}(P_{DR}', r^* k_{2i}' c P_1)^{x_{DO}' \eta_i'}\right)^{\frac{1}{r^* k_{2i}' \eta_i'}} \\
&= \hat{e}(P_{DR}', x_{DO}' c P_1) = \hat{e}(P_1, P_1)^{abc}.
\end{aligned} \tag{23}$$

Now, suppose $\mathscr{B}$ can break the CBDH assumption with advantage ε', $\mathscr{A}_1$ can make at most q_{H_1}, q_{E_1}, q_{E_2}, q_C and q_T times queries to H_1-query, Extract-PPK query, Request-public-key query, Ciphertext query and Trapdoor query, respectively, then

$$\begin{aligned}
&\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4 \wedge \neg E_5] \\
&= \left(1 - \frac{1}{q_{H_1}}\right)^{q_{E_1}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{E_2}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_C} \left(1 - \frac{1}{q_{H_1}}\right)^{q_T} \frac{1}{q_{H_1}} \\
&= \left(1 - \frac{1}{q_{H_1}}\right)^{q_{E_1} + q_{E_2} + q_C + q_T} \frac{1}{q_{H_1}}.
\end{aligned} \tag{24}$$

Since

$$\begin{aligned}\Pr[b=b'] &= \Pr[b=b'|E_6]\Pr[E_6]+\Pr[b=b'|\neg E_6]\Pr[\neg E_6]\\ &\le \Pr[b=b'|E_6]\Pr[E_6]+\Pr[\neg E_6]\\ &= \frac{1}{2}\Pr[E_6]+\Pr[\neg E_6]=\frac{1}{2}+\frac{1}{2}\Pr[\neg E_6],\end{aligned} \tag{25}$$

and

$$\Pr[b=b'] \ge \Pr[b=b'|E_6]\Pr[E_6]=\frac{1}{2}-\frac{1}{2}\Pr[\neg E_6], \tag{26}$$

we have $\Pr[\neg E_6] \ge 2\varepsilon$. Combing with (24), we get Eq. (16) and the lemma is proved.

Lemma 5.2. In the random oracle model, for any PPT adversary $\mathscr{A}_2$, there is an algorithm $\mathscr{B}$ that can break the CBDH assumption with advantage

$$\varepsilon' \ge \frac{2\varepsilon}{q_{H_1}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{E_1}+q_C+q_T} \tag{27}$$

if $\mathscr{A}_2$ wins Game 2 with advantage ε.

Proof. Similar with Lemma 5.1, given an instance of the CBDH assumption $(P_1, aP_1, bP_1, cP_1) \in \mathbb{G}_1^4$, $\mathscr{B}$ calculates the value $\hat{e}(P_1,P_1)^{abc}$ by taking $\mathscr{A}_2$ as a subroutine as follows:

- *Setup*. $\mathscr{B}$ generates $Params = \{\mathbb{G}_1, \mathbb{G}_2, \hat{e}, q, P_1, P_2 = \alpha P_1, Q, P_{pub}, H_i(1 \le i \le 6)\}$ and $P_{mas} = s \in \mathbb{Z}_q^*$, sets $P_{DO} = aP_1$, $P_{DR} = bP_1$, $PK_{CS} = (P_{CS}, R_{CS})$ and $SK_{CS} = (x_{CS}, d_{CS})$, and chooses $ID_I (1 \le I \le q_{H_1})$ randomly as the challenge identity. Finally, it responds both $(Params,\ P_{DO},\ P_{DR},\ PK_{CS},\ SK_{CS})$ and $P_{mas} = s$ to $\mathscr{A}_2$.
- *Phase 1*. $\mathscr{A}_2$ preforms a series of queries with polynomially many times adaptively, they are

– *Hash queries*. $\mathscr{A}_2$ can query $H_i (1 \le i \le 6)$ random oracles. $\mathscr{B}$ responds them as same as Lemma 5.1.

– *Request-public-key query*. $\mathscr{B}$ maintains a list $L_{E_2} = \{\langle ID_i, x_i, y_i, P_i, Y_i\rangle\}$ to respond $\mathscr{A}_2$ for the public key of ID_i. The interaction is the same as *Phase 1* in Lemma 5.1.

– *Extract-private-key query*. When the identity ID_i is queried by $\mathscr{A}_2$, $\mathscr{B}$ first checks whether $ID_i = ID_I$. If not, it performs as follows: if ID_i already exists on L_{H_1} and L_{E_2} in the corresponding tuples $\langle ID_i, \lambda_i, Q_i\rangle$ and $\langle ID_i, x_i, y_i, P_i, Y_i\rangle$, then $\mathscr{B}$ responds to $\mathscr{A}_2$ as $SK_i = (x_i, y_i, sQ_i)$, otherwise performs H_1 *query* and *Request-public-key query* with ID_i and retrieves the corresponding $SK_i = (x_i, y_i, sQ_i)$. If $ID_i = ID_I$, it aborts and denotes this event as E_1.

– *Ciphertext query*. Upon $\mathscr{B}$ receiving a ciphertext query about $(w', ID'_{DO}, ID'_{DR}, ID'_{CS})$. It first checks whether $ID'_{DR} = ID_I$, if not, selects $r' \leftarrow \mathbb{Z}_q^*$ randomly, extracts the corresponding value and computes

$$C'_1 = r'P_2, \tag{28}$$

$$C'_2 = \hat{e}(P'_{DR}, \beta' P'_{CS} + R'_{CS} + \alpha'_{CS} P_{pub})^{r' x'_{DO} \eta'_i}, \tag{29}$$

$$C'_3 = r' k'_{2i} P_1, \tag{30}$$

$$V' = sQ'_{DO} + (r' k'_{2i} + x'_{DO}) R'_{CS}, \tag{31}$$

where $k'_{2i} = H_5(\hat{e}(Q'_{DO}, Q'_{DR})^s)$. Otherwise aborts and denotes the event as E_2. Finally, it outputs $C'_w = (C'_1, C'_2, C'_3, V')$.

– *Trapdoor query*. No matter $\mathscr{A}_2$ makes a data receiver or a data owner trapdoor query. $\mathscr{B}$ performs the follows steps. Upon receiving $(w', ID'_{DO}, ID'_{DR}, ID'_{CS})$, $\mathscr{B}$ aborts and denote this event as E_3 if and only if $ID'_{DR} = ID_I$. Otherwise it selects $t' \leftarrow \mathbb{Z}_q^*$ randomly, extracts the corresponding value and computes

$$T_1 = t'(\beta' P'_{CS} + R'_{CS} + \alpha'_{CS} P_{pub}), \tag{32}$$

$$T_2 = t' k'_{2i} P_1 + x'_{DR} \eta'_i P'_{DO}, \tag{33}$$

where $k'_{2i} = H_5(\hat{e}(Q'_{DO}, Q'_{DR})^s)$. Finally, it outputs $T'_w = (T'_1, T'_2)$ to $\mathscr{A}_2$.

- *Challenge*. After $\mathscr{A}_2$ finished the *Phase 1* queries, it selects $\mathbf{w}_0^*, \mathbf{w}_1^*$ as the challenge keywords sets not be queried before, and sends them to $\mathscr{B}$ with the challenge identity ID_I. $\mathscr{B}$ checks whether $ID'_{DR} = ID_I$, if not, it aborts and denotes this event as E_4, then gives a random bits b' as guess of b. Otherwise chooses a random bit $b \in \{0, 1\}$ and computes ciphertext $C^*_{\mathbf{w}_b} = (C_1^*, C_2^*, C_3^*, V^*)$ as follows. First, pick a random number $r^* \leftarrow \mathbb{Z}_q^*$, compute $C_1^* = r^* c P_1$, $C_2^* = \hat{e}(P'_{DR}, C_1^*)^{x'_{DO}\eta'_i}$, $C_3^* \in \mathbb{G}_1$ and $V^* \in \mathbb{G}_1$. Finally, respond $C^*_{\mathbf{w}_b}$ to $\mathscr{A}_1$.
- *Phase 2*. This phase is same as *Game 2*, $\mathscr{A}_2$ is not allowed to make ciphertext query and two trapdoor queries on $\mathbf{w}_0^*$ and $\mathbf{w}_1^*$. Denote this event as E_5.
- *Guess*. Finally, $\mathscr{A}_2$ outputs its guess $b' \in \{0, 1\}$ on b, meanwhile, $\mathscr{B}$ retrieves $C^*_{\mathbf{w}_b} = (C_1^*, C_2^*, C_3^*, V^*)$ and computes $\left(C_2^*\right)^{\frac{1}{r^*\eta'_i}}$. If $\mathscr{A}_2$ is correct, then no matter $b = 0$ or $b = 1$, it can compute $C_2^* = \hat{e}(P'_{DR}, C_1^*)^{x'_{DO}\eta'_i}$. Furthermore, for unknown $a, b, c \in \mathbb{Z}_q^*$, we set $P'_{DO} = aP_1$, $P'_{DR} = bP_1$ and $P_2 = cP_1$, $\mathscr{B}$ can compute $\hat{e}(P_1, P_1)^{abc}$ as

$$\begin{aligned}\left(C_2^*\right)^{\frac{1}{r^*\eta'_i}} &= \left(\hat{e}(P'_{DR}, C_1^*)^{x'_{DO}\eta'_i}\right)^{\frac{1}{r^*\eta'_i}} \\ &= \left(\hat{e}(P'_{DR}, r^* c P_1)^{x'_{DO}\eta'_i}\right)^{\frac{1}{r^*\eta'_i}} \\ &= \hat{e}(P'_{DR}, x'_{DO} c P_1) = \hat{e}(P_1, P_1)^{abc}.\end{aligned} \tag{34}$$

Now, suppose $\mathscr{B}$ breaks the CBDH assumption with advantage ε', $\mathscr{A}_2$ can make at most q_{H_1}, q_{E_1}, q_C and q_T times queries to H_1-query, Request-public-key query, Ciphertext query and Trapdoor query, respectively, thus,

$$\begin{aligned}&\Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \\ &= \left(1 - \frac{1}{q_{H_1}}\right)^{q_{E_1}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_C} (1 - \frac{1}{q_{H_1}})^{q_T} \frac{1}{q_{H_1}} \\ &= \left(1 - \frac{1}{q_{H_1}}\right)^{q_{E_1}+q_C+q_T} \frac{1}{q_{H_1}}.\end{aligned} \tag{35}$$

Since

$$\begin{aligned}\Pr[b=b'] &= \Pr[b=b'|E_5]\Pr[E_5]+\Pr[b=b'|\neg E_5]\Pr[\neg E_5]\\ &\le \Pr[b=b'|E_5]\Pr[E_5]+\Pr[\neg E_5]\\ &= \frac{1}{2}\Pr[E_5]+\Pr[\neg E_5]=\frac{1}{2}+\frac{1}{2}\Pr[\neg E_5].\end{aligned} \tag{36}$$

and

$$\Pr[b=b'] \ge \Pr[b=b'|E_5]\Pr[E_5]=\frac{1}{2}-\frac{1}{2}\Pr[\neg E_5]. \tag{37}$$

we have $\Pr[\neg E_5] \ge 2\varepsilon$, so combing with (35), we get Eq. (27) and the lemma is proved.

Theorem 5.2 MTP security. In the random oracle model, our CL-BSE scheme achieves semantically MTP security against inside multi-keywords guessing attacks under the CBDH hardness assumption.

The proof of Theorem 5.2 can be achieved by the following two lemmas:

Lemma 5.3. In the random oracle model, for any PPT adversary $\mathscr{A}_1$, there is an algorithm $\mathscr{B}$ can break the CBDH assumption with advantage

$$\varepsilon' \ge \frac{2\varepsilon}{q_{H_1}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{E_1}+q_{E_2}+q_C+q_T} \tag{38}$$

if $\mathscr{A}_1$ wins Game 3 with advantage ε.

Proof. The interaction process in the proof is basically the same as Lemma 5.1 except the *Challenge* phase and the *Guess* phase. They are

- *Challenge*. $\mathscr{A}_1$ still select two sets $\mathbf{w}_0^*, \mathbf{w}_1^*$ as the challenge keywords set and the challenge identity ID_I, sends them to $\mathscr{B}$, $\mathscr{B}$ checks whether $ID'_{DR} = ID_I$, if not, then $\mathscr{B}$ gives a random bits b' as a guess of b. Otherwise, it chooses a random bit $b \in \{0,1\}$ and a random number $t^* \leftarrow \mathbb{Z}_q^*$, then computes $T_1^* = t^*(\beta' P'_{CS} + R'_{CS} + \alpha'_{CS} P_{pub})$, $T_2^* = t^* k'_{2i} P_1 + x'_{DR}\eta'_i P'_{DO}$. Finally, it responds $\mathscr{A}_1$ with $T^*_{\mathbf{w}_b} = (T_1^*, T_2^*)$.
- *Guess*. Finally, $\mathscr{A}_1$ outputs its guess $b' \in \{0,1\}$ on b, and $\mathscr{B}$ chooses $T^*_{\mathbf{w}_b} = (T_1^*, T_2^*)$ to compute $\left(\frac{\hat{e}(T_2^*, P_{pub})}{\hat{e}(P_1, P_{pub})^{t^* \cdot k'_{2i}}}\right)^{\frac{1}{\eta'_i}}$. If $\mathscr{A}_1$ is correct, then no matter $b=0$ or $b=1$, $T_2 = t^* k'_{2i} P_1 + x'_{DR}\eta'_i P'_{DO}$ can be computed. Furthermore, for unknown $a, b, c \in \mathbb{Z}_q^*$, we set $P'_{DO} = aP_1$, $P'_{DR} = bP_1$ and $P_{pub} = cP_1$, $\mathscr{B}$ computes $\hat{e}(P_1, P_1)^{abc}$ as

$$\begin{aligned}&\left(\frac{\hat{e}(T_2^*, P_{pub})}{\hat{e}(P_1, P_{pub})^{t^* \cdot k'_{2i}}}\right)^{\frac{1}{\eta'_i}}\\ &= \left(\frac{\hat{e}(t^* k'_{2i} P_1 + x'_{DR}\eta'_i P'_{DO}, P_{pub})}{\hat{e}(t^* k'_{2i} P_1, P_{pub})}\right)^{\frac{1}{\eta'_i}}\\ &= \left(\hat{e}(x'_{DR}\eta'_i P'_{DO}, P_{pub})\right)^{\frac{1}{\eta'_i}}\\ &= \hat{e}(x'_{DR} P'_{DO}, P_{pub}) = \hat{e}(P_1, P_1)^{abc}.\end{aligned} \tag{39}$$

The analysis process of the advantages ε' that $\mathscr{B}$ computes the above problem is also same as Lemma 5.1, that is Eq. (38) holds and the lemma is proved.

Lemma 5.4. In the random oracle model, for any PPT adversary $\mathscr{A}_2$, there is an algorithm $\mathscr{B}$ can break the CBDH assumption with advantage

$$\varepsilon' \geq \frac{2\varepsilon}{q_{H_1}}\left(1-\frac{1}{q_{H_1}}\right)^{q_{E_1}+q_C+q_T} \tag{40}$$

if $\mathscr{A}_2$ wins Game 4 with advantage ε.

Proof. The interaction process in the proof is basically the same as Lemma 5.2 except the *Challenge* phase and the *Guess* phase. They are

- *Challenge*. $\mathscr{A}_2$ still selects $\mathbf{w}_0^*$, $\mathbf{w}_1^*$ as the challenge keywords set and sends them to $\mathscr{B}$ together with the challenge identity ID_I. $\mathscr{B}$ checks whether $ID'_{DR} = ID_I$, if not, $\mathscr{B}$ aborts and gives a random bits b' as a guess of b. Otherwise chooses a random bit $b \in \{0,1\}$ and computes trapdoor $T^*_{\mathbf{w}_b} = (T_1^*, T_2^*)$ as follows. First, pick a random number $t^* \leftarrow \mathbb{Z}_q^*$, and then compute $T_1^* = t^*(\beta' P'_{CS} + R'_{CS} + \alpha'_{CS} P_{pub}), T_2^* = t^* k'_{2i} P_1 + x'_{DR}\eta'_i P'_{DO}$. Finally, respond to $\mathscr{A}_1$ with $T^*_{\mathbf{w}_b} = (T_1^*, T_2^*)$.
- *Guess*. Finally, $\mathscr{A}_2$ outputs its guess $b' \in \{0,1\}$ on b, meanwhile, $\mathscr{B}$ retrieves $T^*_{\mathbf{w}_b} = (T_1^*, T_2^*)$ and computes $\left(\frac{\hat{e}(T_2^*, T_1^*)}{\hat{e}(t^* \cdot k'_{2i} \cdot P_1, T_1^*)}\right)^{\frac{1}{t^*\eta'_i(\beta' x'_{CS} + d'_{CS})}}$. If $\mathscr{A}_2$ is correct, then no matter $b=0$ or $b=1$, algorithm $\mathscr{B}$ can compute $T_1^* = t^*(\beta' P'_{CS} + R'_{CS} + \alpha'_{CS} P_{pub})$, $T_2^* = t^* k'_{2i} P_1 + x'_{DR}\eta'_i P'_{DO}$. Furthermore, for unknown $a, b, c \in \mathbb{Z}_q^*$, we set $P'_{DO} = aP_1$, $P'_{DR} = bP_1$, and $P_2 = cP_1$, $\mathscr{B}$ can compute $\hat{e}(P_1, P_1)^{abc}$ as following:

$$\begin{aligned}
&\left(\frac{\hat{e}(T_2^*, T_1^*)}{\hat{e}(t^* k'_{2i} P_1, T_1^*)}\right)^{\frac{1}{t^*\eta'_i(\beta' x'_{CS} + d'_{CS})}} \\
&= \left(\frac{\hat{e}(t^* k'_{2i} P_1 + x'_{DR}\eta'_i P'_{DO}, t^*\left(\beta' x'_{CS} + d'_{CS}\right) P_2)}{\hat{e}(t^* k'_{2i} P_1, t^* \Gamma'_{CS} P_2)}\right)^{\frac{1}{t^*\eta'_i(\beta' x'_{CS} + d'_{CS})}} \\
&= \left(\hat{e}(x'_{DR}\eta'_i P'_{DO}, t^*\left(\beta' x'_{CS} + d'_{CS}\right) P_2)\right)^{\frac{1}{t^*\eta'_i(\beta' x'_{CS} + d'_{CS})}} \\
&= \hat{e}(x'_{DR} P'_{DO}, P_2) = \hat{e}(P_1, P_1)^{abc}.
\end{aligned} \tag{41}$$

The analysis process of the advantages ε' that $\mathscr{B}$ computes the above problem is also same as Lemma 5.2, that is Eq. (40) holds and the lemma is proved.

6 Performance Analysis

In this section, we analyze the performance of our scheme by comparing it with some existing schemes in [4,5,28–30,36,37].

First, we give some basic operations used in the scheme and the executing times of a single operation in Table 2. These times of operations are averaged over 1000 runs on a personal computer (Lenovo with Windows 10 operating system, Intel (R) Core (TM) i7 −7700 CPU @ 3.60 GHz and 8 GB RAM memory) using the Pairing-Based Cryptography (PBC) library [38] in Ubuntu10.

Table 2: Some operations and their overhead time (ms)

Symbol	Description	Times
T_P	Bilinear pairing operation	5.787
T_H	Hash-to-point operation	5.693
T_S	Scalar multiplication	2.355
T_E	Modular exponentiation	0.794
T_h	General hash function	0.005
T_A	Addition operation	0.003

Figs. 2–5 and Table 3 describe the computation overhead of different algorithms in each scheme. Specifically, the computational overhead in the ciphertext generation (Fig. 3) of our scheme is slightly higher than [30,36]. In trapdoor generation process (Fig. 4), the computational overhead of the scheme is higher than [4,5,30] since the enhanced trapdoor privacy and authentication functionality. In test process (Fig. 5), its computational overhead is slightly higher than in [29,30] because our scheme is server-designated, that is, the public/private key pairs of the server are involved in the operation. However, in terms of total time, the time overhead of our new scheme is only slightly higher than [30]. It has some advantages when DO (or DU) retrieves emails and is more in line with practical application scenarios.

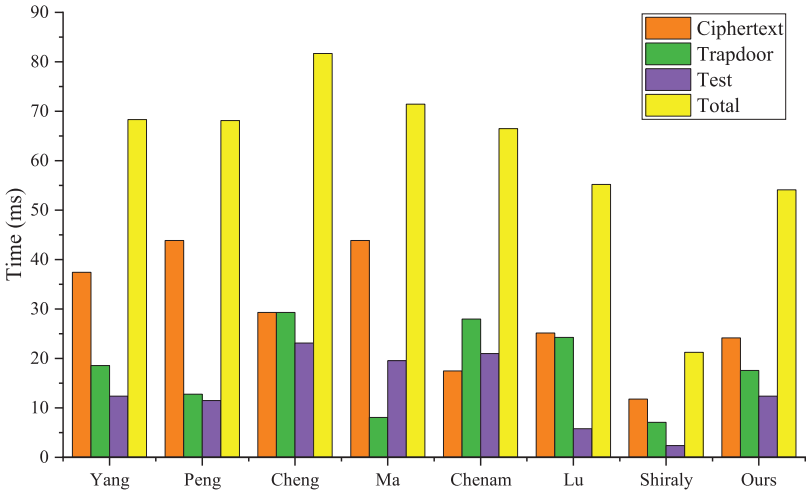

Figure 2: Computation overhead in each phase

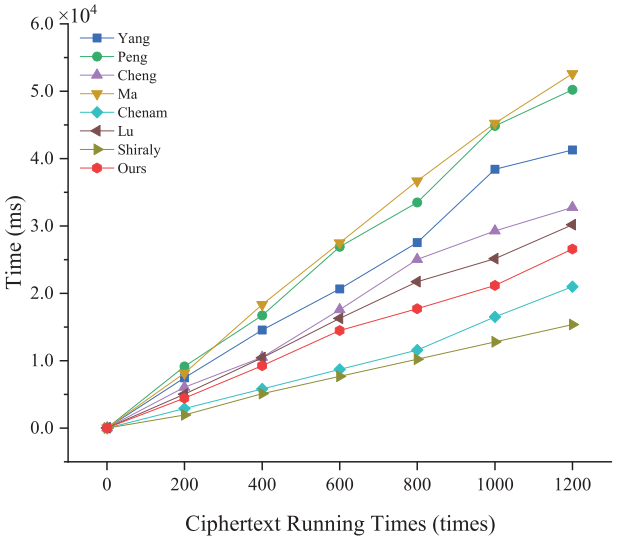

Figure 3: Running time of encryption

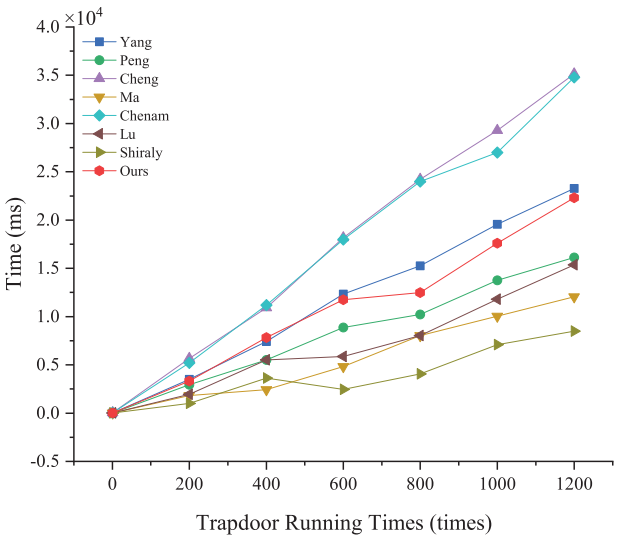

Figure 4: Running time of trapdoor

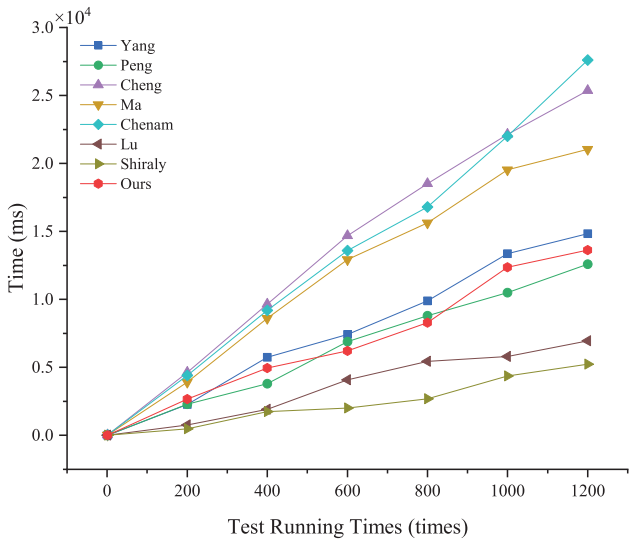

Figure 5: Running time of test

Table 3: The computational overhead of the schemes (ms)

Schemes	Ciphertext	Trapdoor	Test
Yang et al. [28]	$3T_P + 2T_H + 3T_S + 2T_E + 3T_h + T_A$	$T_P + T_H + 3T_S + 3T_h + 2T_A$	$2T_P + T_E + T_A$
Peng et al. [5]	$3T_P + 3T_H + 4T_S + 2T_h$	$T_H + 3T_S + T_h + 2T_A$	$T_P + T_S + T_h + 4T_A$
Cheng et al. [37]	$T_H + 10T_S + 5T_h + 5T_A$	$T_H + 10T_S + 5T_h + 5T_A$	$4T_P$
Ma et al. [4]	$3T_P + 3T_H + 4T_S + T_h + T_A$	$T_H + T_S + T_A$	$T_P + 2T_H + T_S + T_h + 2T_A$

(Continued)

Table 3 (continued)

Schemes	Ciphertext	Trapdoor	Test
Chenam et al. [36]	$T_H + 5T_S + 3T_h + 3T_A$	$T_P + T_H + 7T_S + 3T_h + 4T_A$	$2T_P + 4T_S + T_h + 2T_A$
Lu et al. [29]	$2T_P + T_H + 3T_S + T_E + 3T_h + T_A$	$T_P + 2T_H + 3T_S + 3T_h + T_A$	$T_P + T_h$
Shiraly et al. [30]	$5T_S + 3T_h + T_A$	$3T_S + 2T_h + T_A$	$T_S + T_h$
Ours	$2T_P + 5T_S + T_E + 5T_h + 2T_A$	$T_P + 5T_S + 5T_h + 3T_A$	$2T_P + T_E$

Subsequently, we make a comparison in terms of communication costs, including the size of public key $|PK|$, ciphertext $|CT|$ and trapdoor $|TD|$, which are presented in Table 4. In the table, the notations $|\mathbb{G}_1|$, $|\mathbb{G}_2|$ and $|\mathbb{Z}_q|$ denote the bit length size for each element in $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{Z}_q$, respectively. It is clearly see that the size of ciphertext of our scheme is the same as Yang et al.'s scheme [28] and is smaller than Cheng et al.'s scheme [37], a sightly larger than other schemes; same as Yang et al.'s scheme [28], the size of trapdoor of our scheme is smaller than other schemes except Ma et al.'s scheme [4].

Table 4: The communication overhead of the schemes (bits)

	Yang et al. [28]	Peng et al. [5]	Cheng et al. [37]	Ma et al. [4]	Chenam et al. [36]	Lu et al. [29]	Shiraly et al. [30]	Ours
$\|PK\|$	$2\|\mathbb{G}_1\|$	$4\|\mathbb{G}_1\|$	$4\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$4\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$	$4\|\mathbb{G}_1\|$	$4\|\mathbb{G}_1\|$
$\|CT\|$	$2\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$\|\mathbb{G}_1\|+\|\mathbb{Z}_q^*\|$	$4\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|+\|\mathbb{Z}_q^*\|$	$2\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|+\|\mathbb{Z}_q^*\|$	$2\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$
$\|TD\|$	$2\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|$	$4\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$	$\|\mathbb{Z}_q^*\|$	$\|\mathbb{Z}_q^*\|$	$2\|\mathbb{G}_1\|$

Finally, we present some additional performance comparisons in the Table 5. In the table, SCF denotes designated server test, AUT denotes authenticated function, BSE denotes bidirectional searchable encryption and ASSUM denotes the difficulty assumption of the scheme security depends on. Finally, we find that our scheme is a certificateless authenticated bidirectional searchable encryption scheme with a designated server test that achieves both MCI and MTP security under the CBDH hardness assumption.

Table 5: Other performance comparison

Schemes	DST	AUT	BSE	MCI	MTP	ASSUM
Yang et al. [28]	✓	✓	×	✓	✓	CBDH
Peng et al. [5]	×	×	×	×	×	BDH & GBDH
Cheng et al. [37]	✓	✓	×	✓	✓	CODH
Ma et al. [4]	✓	×	×	✓	×	BDH
Chenam et al. [36]	✓	✓	×	×	×	CBDH & DBDH
Zhang et al. [13]	×	✓	✓	✓	×	BDH
Lu et al. [29]	×	✓	×	×	×	BDH

(Continued)

Table 5 (continued)

Schemes	DST	AUT	BSE	MCI	MTP	ASSUM
Shiraly et al. [30]	×	✓	×	×	×	DDH & GDH
Ours	✓	✓	✓	✓	✓	CBDH

7 Conclusion

Based on the certificateless public key authenticated encryption with keyword search (CL-PAEKS) cryptosystem and the bidirectional searchable functionality, this paper proposed a new cryptographic approach named blockchain-based certificateless authenticated bidirectional searchable encryption (CL-BSE). To some extent, it can be regarded as avoiding the key escrow and certificate management problem in the PEBKS scheme, and can also be considered as appending distinctive features which allow a data owner to retrieve the keyword ciphertext from server in the CL-PAEKS cryptosystem. Taking the cloud email system as the actual application scenario, we build a concrete construction of the CL-BSE scheme. The security analysis of the scheme indicates that it can achieve both MCI-secure and MTP-secure against IKGA under the CBDH hardness assumption.

Acknowledgement: The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

Funding Statement: This work was supported by the National Natural Science Foundation of China (Nos. 62172337, 62241207) and Key Project of Gansu Natural Science Foundation (No. 23JRRA685).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Y. Sun, X. Du; data collection: Y. Sun; analysis and interpretation of results: Y. Sun, X. Du, X. Yang; draft manuscript preparation: Y. Sun, S. Niu. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Song, D., Wagner, D., Perrig, A. (2000). Practical techniques for searching on encrypted data. *IEEE Symposium on Security and Privacy*, pp. 44–55. Berkeley, USA.
2. Boneh, D., di Crescenzo, G., Ostrovsky, R., Persiano, G. (2004). Public key encryption with keyword search. *International Conference on the Theory and Applications of Cryptographic Techniquespp*, pp. 506–522. Interlaken, Switzerland.
3. Baek, J., Safavi-Naini, R., Susilo, W. (2008). Public key encryption with keyword search revisited. *International Conference on Computational Science and its Applications*, pp. 1249–1259. Perugia, Italy.
4. Ma, M., He, D., Kumar, N., Choo, K. K. R., Chen, J. (2018). Certificateless searchable public key encryption scheme for industrial Internet of Things. *IEEE Transactions on Industrial Informatics, 14(2),* 759–767.

5. Peng, Y., Cui, J., Peng, C., Ying, Z. (2014). Certificateless public key encryption with keyword search. *China Communications, 11(11),* 100–113.
6. Byun, J. W., Rhee, H. S., Park, H., Lee, D. H. (2006). Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. *Third VLDB Workshop on Secure Data Management*, pp. 75–83. Seoul, Korea.
7. Huang, Q., Li, H. (2017). An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Information Sciences, 403(C),* 1–14.
8. Qin, B., Chen, Y., Huang, Q., Liu, X., Zheng, D. (2020). Public-key authenticated encryption with keyword search revisited: Security model and constructions. *Information Sciences, 516,* 515–528.
9. Pan, X., Li, F. (2021). Public-key authenticated encryption with keyword search achieving both multi-ciphertext and multi-trapdoor indistinguishability. *Journal of Systems Architecture, 115(C),* 102075.
10. Al-Riyami, S. S., Paterson, K. G. (2003). Certificateless public key cryptography. *9th International Conference on the Theory and Application of Cryptology and Information Security*, pp. 452–473. Taipei, Taiwan.
11. He, D., Ma, M., Zeadally, S., Kumar, N., Liang, K. (2018). Certificateless public key authenticated encryption with keyword search for industrial internet of things. *IEEE Transactions on Industrial Informatics, 14(8),* 3618–3627.
12. Yang, X., Wen, H., Liu, L., Ren, N., Wang, C. (2023). Blockchain-enhanced certificateless signature scheme in the standard model. *Mathematical Biosciences and Engineering, 20(7),* 12718–12730.
13. Zhang, W., Qin, B., Dong, X., Tian, A. (2021). Public-key encryption with bidirectional keyword search and its application to encrypted emails. *Computer Standards & Interfaces, 78,* 103542.
14. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security, 19(5),* 895–934.
15. Kamara, S., Papamanthou, C. (2013). Parallel and dynamic searchable symmetric encryption. *International Conference on Financial Cryptography and Data Security*, pp. 258–274. Okinawa, Japan.
16. Li, J., Huang, Y., Wei, Y., Lv, S., Liu, Z. et al. (2021). Searchable symmetric encryption with forward search privacy. *IEEE Transactions on Dependable and Secure Computing, 18(1),* 460–474.
17. Rhee, H. S., Park, J. H., Susilo, W., Lee, D. H. (2010). Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software, 83(5),* 763–771.
18. Chen, R., Mu, Y., Yang, G., Guo, F., Huang, X. et al. (2016). Server-aided public key encryption with keyword search. *IEEE Transactions on Information Forensics and Security, 11(12),* 2833–2842.
19. Hu, C., Liu, P. (2012). An enhanced searchable public key encryption scheme with a designated tester and its extensions. *Journal of Computers, 7(3),* 716–723.
20. Tang, Q., Chen, L. (2010). Public-key encryption with registered keyword search. *Proceedings of the 6th European Workshop on Public Key Infrastructures, Services and Applications*, pp. 163–178. Pisa, Italy.
21. Wu, T. Y., Chen, C. M., Wang, K. H., Pan, J. S., Zheng, W. et al. (2018). Security analysis of Rhee et al.'s public encryption with keyword search schemes: A review. *Journal of Network Intelligence, 3(1),* 16–25.
22. Yau, W. C., Phan, R. C. W., Heng, S. H., Goi, B. M. (2013). Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester. *International Journal of Computer Mathematics, 90(12),* 2581–2587.
23. Noroozi, M., Eslami, Z. (2019). Public key authenticated encryption with keyword search: Revisited. *IET Information Security, 13(4),* 336–342.
24. Cheng, L. X., 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". *Journal of Systems Architecture, 119,* 102248.
25. Fuhr, T., Paillier, P. (2007). Decryptable searchable encryption. *Proceedings of the First International Conference on Provable Security*, pp. 228–236. Wollongong, Australia.

26. Hofheinz, D., Weinreb, E. (2008). Searchable encryption with decryption in the standard model. *Cryptology ePrint Archive*.
27. Wu, T. Y., Chen, C. M., Wang, K. H., Wu, J. M. T. (2019). Security analysis and enhancement of a certificateless searchable public key encryption scheme for IIoT environments. *IEEE Access, 7,* 49232–49239.
28. Yang, G., Guo, J., Han, L., Liu, X., Tian, C. (2022). An improved secure certificateless public-key searchable encryption scheme with multi-trapdoor privacy. *Peer-to-Peer Networking and Applications, 15(1),* 503–515.
29. Lu, Y., Li, J. (2019). Constructing certificateless encryption with keyword search against outside and inside keyword guessing attacks. *China Communications, 16(7),* 156–173.
30. Shiraly, D., Pakniat, N., Noroozi, M., Eslami, Z. (2022). Pairing-free certificateless authenticated encryption with keyword search. *Journal of Systems Architecture, 124,* 102390.
31. Boneh, D., Franklin, M. (2003). Identity-based encryption from the weil pairing. *SIAM Journal on Computing, 32(3),* 586–615.
32. Barreto, P. S. L. M., Kim, H. Y., Lynn, B., Scott, M. (2002). Efficient algorithm for pairing-based cryptosystems. *22nd Annual International Cryptology Conference Santa Barbara*, pp. 354–369. California, USA.
33. Joux, A. (2002). The weil and tate pairing as building blocks for public key cryptosystems. *5th International Algorithmic Number Theory Symposium*, pp. 20–32. Sydney, Australia.
34. Han, M., Xu, P., Xu, L., Xu, C. (2022). TCA-PEKS: Trusted certificateless authentication public-key encryption with keyword search scheme in cloud storage. *Peer-to-Peer Network Appliction, 16(1),* 156–169.
35. Uwizeye, E., Wang, J., Cheng, Z., Li, F. (2019). Certificateless public key encryption with conjunctive keyword search and its application to cloud-based reliable smart grid system. *Annals of Telecommunications, 74(7),* 435–449.
36. Chenam, V. B., Ali, S. T. (2022). A designated cloud server-based multi-user certificateless public key authenticated encryption with conjunctive keyword search against IKGA. *Computer Standards & Interfaces, 81(C),* 103603.
37. Cheng, L. X., Meng, F. (2023). Certificateless public key authenticated searchable encryption with enhanced security model in IIoT applications. *IEEE Internet of Things Journal, 10(2),* 1391–1400.
38. PBC Library: The pair-based cryptography library. http://crypto.stanford.edu/pbc/ (accessed on 06/01/2023).