The ISC Int'l Journal of Information Security

ISeCure

November 2023, Volume 15, Number 3 (pp. 1–17)

http://www.isecure-journal.org

PRESENTED AT THE ISCISC'2023 IN TEHRAN, IRAN.

Designated-Server Hierarchical Searchable Encryption in Identity-Based Setting **

Danial Shiraly [1], Nasrollah Pakniat [2], and Ziba Eslami [1,*]

[1] *Department of Computer and Data Science, Shahid Beheshti University, Tehran, Iran*

[2] *Information Science Research Center, Iranian Research Institute for Information Science and Technology (IranDoc), Tehran, Iran*

A R T I C L E I N F O.

Keywords:
Cloud Computing, Hierarchical Identity-Based Searchable Encryption, Identity-Based Encryption, Searchable Encryption

Type:
Research Article

doi: 10.22042/isecure.2023.416676.1014

A B S T R A C T

Public key encryption with keyword search (PEKS) is a cryptographic primitive designed for performing secure search operations over encrypted data stored on untrusted cloud servers. However, in some applications of cloud computing, there is a hierarchical access-privilege setup among users so that upper-level users should be able to monitor data used by lower-level ones in the hierarchy. To support such situations, Wang *et al.* introduced the notion of hierarchical ID-based searchable encryption. However, Wang *et al.*'s construction suffers from a serious security problem. To provide a PEKS scheme that securely supports hierarchical structures, Li *et al.* introduced the notion of hierarchical public key encryption with keyword search (HPEKS). However, Li *et al.*'s HPEKS scheme is established on traditional public key infrastructure (PKI) which suffers from costly certificate management problem. To address these issues, in this paper, we consider designated-server HPEKS in identity-based setting. We introduce the notion of designated-server hierarchical identity-based searchable encryption (dHIBSE) and provide a formal definition of its security model. We then propose a dHIBSE scheme and prove its security under our model. Finally, we provide performance analysis as well as comparisons with related schemes to show the overall superiority of our dHIBSE scheme.

1 Introduction

With the enormous benefits of cloud computing, organizations and individual users prefer to outsource their data from local to cloud storage. However, due to the untrusted nature of the cloud servers, it is necessary to somehow ensure the confidentiality and privacy of sensitive data. To achieve such a guarantee, users can encrypt their data before sending them to the cloud service provider. Nevertheless, traditional encryption techniques make sharing and searching over outsourced data a daunting task. To overcome this issue, in 2000, Song *et al.* [1] introduced the notion of Searchable Encryption (SE) which provides

* Corresponding author.
**The ISCISC'2023 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: d_shiraly@sbu.ac.ir, Pakniat@irandoc.ac.ir, z_eslami@sbu.ac.ir

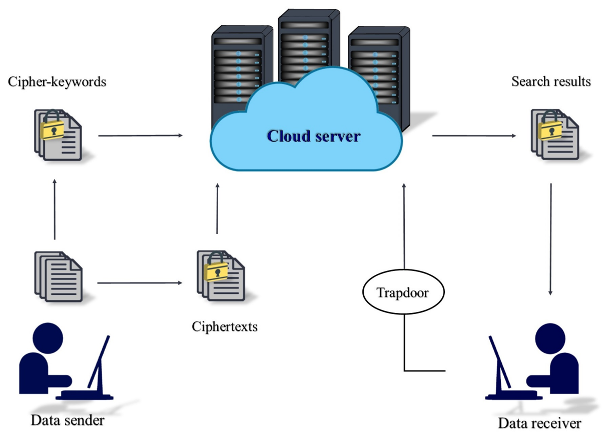

Figure 1. The general framework of SE

data confidentiality and at the same time preserves search-ability on the server side. It equips users with trapdoors that allow them to search over ciphered data for keywords, without decryption. In the basic setup, an SE scheme contains three types of entities, as follows:

Data sender, a cloud service user who wants to outsource a collection of its documents and their associated keywords. Therefore, the data sender needs to generate "searchable" ciphertexts corresponding to these keywords, encrypt the collection of documents, and finally upload the results on the cloud server.

Data receiver, the entity that can search and retrieve documents outsourced on the cloud server containing desired keywords. To do so, it has to generate suitable trapdoors for its query keywords and send them to the cloud server.

Cloud server which manages encrypted collections of documents uploaded by data senders. Upon receiving a valid trapdoor, it can search on behalf of the data receiver to find documents containing the corresponding keywords. During the process of searching, the cloud server remains unable to decrypt the contents of documents or keywords.

The general framework of a searchable encryption scheme is shown in Figure 1. Song *et al.*'s protocol is in the symmetric setting which makes it mainly suitable for the data sender itself (i.e. data owner) to search over ciphertexts. The massive amount of data generated and outsourced by governments, enterprises, and firms calls for SE in the public key setting. This was accomplished by Boneh *et al.* [2] who introduced a cryptographic notion called Public Key Encryption with Keyword Search, or PEKS for short. PEKS schemes consist of 3 algorithms: **IndexGen** through which, a data sender encrypts the keywords by using the data receiver's public key and uploads the results on the cloud server; **TrapdoorGen** through which, the receiver generates a trapdoor corresponding to a keyword by using its private key and sends the result to the cloud server; **Test** through which, the cloud server checks if a searchable ciphertext and a trapdoor are corresponded to the same keyword by using the data receiver's public key.

The fact is that large multi-receiver setups (such as enterprises and firms) that benefit most from searchable encryption technology, usually support a hierarchical structure among their users (employees). In such settings, superiors who have higher access privileges should be able to monitor their lowers. In other words, in addition to the intended receiver, the encrypted messages should also be searchable by his/her superiors.

To integrate support for hierarchical access control requirements into PEKS, Wang *et al.* [3] introduced the notion of hierarchical ID-based encryption with keyword search (HIBEKS) and proposed a HIBEKS scheme. In the HIBEKS framework, the identity of a user is represented as a hierarchical tree structure, and a high-level user is able to compute valid trapdoors for searching over ciphertexts corresponding to its descendant identities. Later, in 2020, Li *et al.* [4] introduced a public key tree structure in SE and defined the notion of designated-server Hierarchical Public Key Encryption with Keyword Search (dHPEKS). In a dHPEKS scheme, each user in a group of data receivers owns a node of a public key tree, which in turn contains its public information. At the top of the hierarchy, the supervisor of the group generates a master secret key. Then, it generates the secret key for itself and its child users. The other users are merely responsible for generating the secret keys of their child users. A data sender can generate the searchable ciphertext for any user in any data receiver group. However, compared to the ordinary dPEKS schemes, in addition to the intended data receiver, any user superior to that receiver, according to the group hierarchical structure, is able to generate a trapdoor and search over that receiver's encrypted data.

1.1 Motivation and Contribution

Being able to search over employees' data is a desirable property that is achieved by introducing hierarchical structures to the notion of searchable encryption by Wang *et al.* [3]. However, 1) Wang *et al.*'s scheme is not secure against an important type of attack called offline keyword guessing attack (KGA) and 2) it imposes some restrictions on the hierarchical level of the system users which conflicts with cloud scalability, an essential property of cloud computing.

Recently, Li *et al.* [4] and Liu *et al.* [5] proposed two designated-tester PEKS schemes that support hierarchical structures. These schemes don't suffer from the drawbacks of Wang *et al.*'s scheme [3]; however, the same as all existing cryptosystems in the traditional public key infrastructure (PKI) setting, the schemes of [4, 5] suffer from certificate management problem.

To simultaneously address the existing issues in the schemes of [3–5], in this paper,

- We define the notion of designated-server hierarchical identity-based searchable encryption (dHIBSE).
- We formalize the security model for dHIBSE.
- We propose a concrete dHIBSE scheme and prove its security in the proposed security model.
- We compare the proposed scheme with the related ones and show its overall superiority.

1.2 Related Works

The notion of PEKS was introduced in 2004 by Boneh *et al.* [2]. The authors also defined a security model and proposed the first PEKS scheme based on bilinear pairing for the multi-user model. Nevertheless, Baek *et al.* [6] point out that Boneh *et al.*'s scheme has the limitation that a secure communication channel between each data receiver and the cloud server is required to transmit the trapdoors. In order to overcome this limitation, the authors enhanced the definition of PEKS and proposed the concept of the "secure channel-free PEKS" (SCF-PEKS). In 2006, Byun *et al.* [7] introduced an important attack against PEKS schemes called Keyword Guessing Attack (KGA), which derives from the fact that in real-world applications, keywords are usually chosen from a low-entropy space. In a KGA, after obtaining a trapdoor, an adversary generates the searchable ciphertext corresponding to each possible keyword and then checks it with the given trapdoor using a publicly available Test algorithm. Therefore, by launching this attack, an adversary may correctly obtain the corresponding keyword to the given trapdoor. In order to ensure security against KGA, in 2010, Rhee *et al.* [8] defined a new security model that improved the security model of [6] and allowed the adversary to obtain the relationship between the trapdoor and the non-challenged ciphertexts. In order to provide the new security property, in their seminal work, Rhee *et al.* [8] introduced the notion of "searchable public key encryption with a designated tester" (dPEKS). In dPEKS schemes, the server is also equipped with a pair of public and private keys, and the public key of the server gets involved in both keyword encryption and trapdoor generation algorithms. Consequently, in a dPEKS scheme, only the server can check if a trapdoor and a ciphertext match or not. This way, dPEKS schemes are secure against KGAs. After Rhee *et al.*'s work, several other dPEKS schemes were also proposed to provide security against KGA and improve the security of dPEKS schemes [9–14].

Unfortunately, all the above-mentioned schemes are constructed based on the PKI setting which suffers from the well-known certificate management problem. To address the certificate management problem in PEKS schemes, Abdalla *et al.* [15] formalized the notion of identity-based encryption with keyword search (IBEKS) in 2005. In an IBEKS scheme, a user's public key can be directly calculated from his identity (such as an email address or phone number), while the user's private keys are generated by a trusted authority, called a private key generator (PKG), according to the user's request. This link between the user's public key and his identity information removes the need of certificates used in traditional PKI setting. After Abdalla *et al.*'s work [15], some other IBEKS schemes were proposed in the literature [16–19]. However, in the absence of a secure channel, these schemes do not provide trapdoor-indistinguishability and consequently suffer from the vulnerability caused by the KGA. In order to provide security against KGA in identity-based setting, in 2014, Wu *et al.* [20] introduced the notion of identity-based encryption with keyword search with a designated tester (dIBEKS) and proposed the first dIBEKS scheme. In 2016, Wang *et al.* [21] proposed two identity-based encryption with keyword search with a designated tester schemes in a peer-to-peer group, where the group users can share and search private data in the cloud. Later, Lu *et al.* [9] proposed a designated server identity-based encryption scheme with conjunctive keyword search (dIBECKS) and claimed that their proposed scheme provides the ciphertext-indistinguishability and the trapdoor-indistinguishability against chosen keyword attacks. Recently, in [22], Noroozi and Eslami analyzed ciphertext and trapdoor security of the dIBEKS scheme of [21] and the dIBECKS scheme of [9]. Furthermore, to ensure security against KGA in other settings, several PEKS schemes are proposed in [23–32], to which the interested readers are referred for a comprehensive study.

Now consider a scenario where a data sender wants to share his documents (to which corresponding cipher-keywords must be attached) with multiple data receivers. The issue is that most of the existing PEKS schemes merely support single-receiver settings. Therefore, a data sender has to encrypt the same keyword multiple times for different data receivers, which increases the computation and storage overhead. To solve this problem, in 2006, Baek *et al.* [33] proposed the first multi-receiver PEKS scheme. Many PEKS schemes have been constructed to support multi-receiver scenarios [21, 34–36]. However, in the large multi-receivers setups, users are usually arranged hierarchically, according to their corresponding roles and responsibilities. In this hierarchical structure, a user with higher access permission should have the advantage of monitoring his lower-level users' data. Therefore, users with higher access permissions should be able to generate valid trapdoors for any keywords, in order to preserve search ability over the encrypted documents corresponding to their lowers. Unfortunately, none of the above-mentioned SE schemes support hierarchical access permission for users. To solve this problem, Wang *et al.* [3] introduced the notion of hierarchical ID-based encryption with keyword search (HIBEKS) and proposed the first HIBEKS scheme. Wang *et al.* claimed that their proposed scheme provides ciphertext-indistinguishability and security against offline KGAs launched by outside adversaries (anyone except the data receiver and the server). However, their construction is completely insecure against offline KGAs from outside adversaries.[1] Another drawback of the HIBEKS framework proposed by Wang *et al.* [3] is that it requires to determine the exact maximum hierarchical level in the system set-up phase. This can be problematic for many business and individual users in real-world applications. In 2020, Li *et al.* [4] introduced the structure of the public key tree, which is a multi-way tree, into the SE. Then, the authors defined the notion of the designated-server hierarchical public key encryption with keyword search (dHPEKS) which uses a public key tree in order to support hierarchical access control. The authors also proposed a dHPEKS scheme and proved that their construction provides ciphertext-indistinguishability against chosen keyword attacks along with trapdoor-indistinguishability against outside adversaries under CDH, DLIN, and DBDH assumptions. Later, in 2021, Liu *et al.* [5] proposed a dHPEKS scheme by using a distributed Two-Trapdoor Public-key Cryptosystem [37]. Their proposed scheme, the same as the dHPEKS scheme of [4], is constructed based on a public key tree of the user's hierarchical structure in the PKI setting. The authors proved the security of their proposed dHPEKS scheme under CDH assumption.

1.3 Paper Organization

The rest of this paper is arranged as follows. We describe some preliminary materials in Section 2. In Section 3, the framework of designated-server hierarchical identity-based searchable encryption (dHIBSE) and the formal definition of security model for dHIBSE are presented. We propose a dHIBSE scheme in Section 4 and analyze its security in Section 5. In Section 6, we make performance analysis and comparisons. Finally, Section 7 concludes the paper.

2 Preliminaries

In this section, a brief review of some preliminary materials is presented. The interested readers are referred to [38] for more details.

Definition 1. A symmetric bilinear group description Γ is a tuple (q, G_1, G_2, e, P) consists of the two cyclic groups G_1 and G_2 of prime order q, a bilinear map $e : G_1 \times G_1 \to G_2$, and P which is a generator of G_1.

Definition 2. Let $\Gamma = (q, G_1, G_2, e, P)$ be a symmetric bilinear group description. The gap bilinear Diffie-Hellman ($GBDH$) assumption is that for any probabilistic polynomial time (PPT) adversary A, the advantage $Adv_{\Gamma}^{GBDH}(A, q_{DBDH})$, defined by the following probability, is negligible.

$$Adv_{\Gamma}^{GBDH}(A, q_{DBDH}) := Pr[T = e(P,P)^{abc} | a, b, c \in Z_q; T \leftarrow A^{O_{DBDH}}(\Gamma, aP, bP, cP)].$$

O_{DBDH} represents a decision bilinear Diffie-Hellman oracle that takes (aP, bP, cP, T) as input and outputs 1 if $T = e(P,P)^{abc}$ holds and 0 otherwise. q_{DBDH} represents the maximum number of queries that A could query from the O_{DBDH} [39, 40].

Definition 3. Let $\Gamma = (q, G_1, G_2, e, P)$ be a symmetric bilinear group description. The computational Diffie-Hellman (CDH) assumption in G_1 is that for any PPT adversary A, the advantage $Adv_{\Gamma}^{CDH}(A)$, defined by the following probability, is negligible.

$$Adv_{\Gamma}^{CDH}(A) := Pr[Q = abP | a, b \in Z_q; Q \leftarrow A(\Gamma, aP, bP)].$$

3 Definition and Security Model of dHIBSE

In this section, we first describe the framework of a dHIBSE scheme and then define its security requirements. In a dHIBSE scheme, each user has an identity that is represented as a hierarchical tree structure,

[1] Using the notations of Wang *et al.* [3], upon receiving a trapdoor $(T_{j_1}, \cdots, T_{j_{l-j+1}}, S_j)$, an outside adversary $\mathcal{A}$ guesses a keyword w and computes $X' = \hat{g}$, $Y' = (\hat{g}'_3 \hat{h}_1^{I_1} \cdots \hat{h}_i^{I_i} \hat{h}^w)$, $Z' = e(g_2, g'_1)$, where $ID_R = [I_1, I_2, \cdots, I_i]$ is the identity of the data receiver. Then, it checks the following equation: $Z' \stackrel{?}{=} \frac{e(T_{j_1} T_{j_1}^{I_{j+1}} \ldots T_{j_{i-j+1}}^{I_i}, X')}{e(S_j, Y')}$. If it holds, then the trapdoor is associated with the keyword w; otherwise, $\mathcal{A}$ guesses another possible keyword. Therefore, this scheme is not secure against offline KGAs performed by outside adversaries.

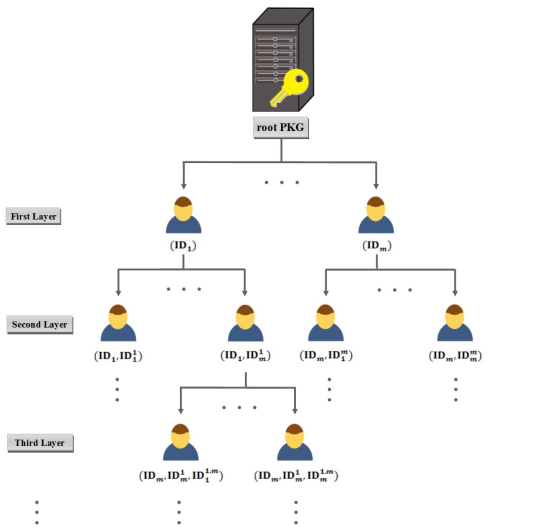
root PKG
First Layer
Second Layer
Third Layer
$(\mathbf{ID}_1)$
$(\mathbf{ID}_m)$
$(\mathbf{ID}_1, \mathbf{ID}_1^1)$
$(\mathbf{ID}_1, \mathbf{ID}_m^1)$
$(\mathbf{ID}_m, \mathbf{ID}_1^m)$
$(\mathbf{ID}_m, \mathbf{ID}_m^m)$
$(\mathbf{ID}_m, \mathbf{ID}_m^1, \mathbf{ID}_1^{1,m})$
$(\mathbf{ID}_m, \mathbf{ID}_m^1, \mathbf{ID}_m^{1,m})$

Figure 2. The general hierarchical tree structure of a dHIBSE scheme

and high-level users are allowed to delegate their key generation ability to low-level users. The root PKG, which sits at the top of the hierarchical structure, is in charge of creating the private keys intended for the users who are positioned within the first layer of the hierarchy. These private keys are then sent to the corresponding users. A user with ID-tuple $ID|_t = (ID_1, ID_2, \cdots, ID_t)$ at level t of the hierarchical tree can compute the private keys of its descendant identities. Consequently, the user is able to generate valid trapdoors for searching over ciphertexts corresponding to him and all his children, not the opposite. Figure 2 depicts the general hierarchical tree structure of a dHIBSE scheme. (For simplicity, in this figure we assume that the root PKG and every user of the system has exactly m child users.)

3.1 The Definition

A dHIBSE scheme consists of six PPT algorithms (*Setup*, *KeyGen*, *ServerKeyGen*, *dHIBSE*, *Trapdoor*, *Test*). The definition of each of these algorithms is provided in the following:

- **Setup**: This algorithm is performed by the root PKG. On input of the security parameter λ, this algorithm outputs the master secret key s, and the public parameters $prms$. The root PKG keeps s secret and publishes $prms$.
- **KeyGen**: This algorithm is performed by the root PKG or a lower-level PKG with ID-tuple $ID|_{t-1} = (ID_1, \cdots, ID_{t-1})$. On input of the public parameters $prms$, ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$ and the private key of the ancestor of $ID|_t$, i.e., $s_{ID|_{t-1}}$, this algorithm outputs $s_{ID|_t}$ which will be sent securely to the user with ID-tuple $ID|_t$. Note that if $t = 1$ then, $s_{ID|_{t-1}}$ would be s.
- **ServerKeyGen**: This algorithm is performed by the server. On input of the public parameters $prms$, this algorithm outputs the secret value x_S, and the public key P_S. The server keeps x_S secret and publishes P_S.
- **dHIBSE**: This algorithm is performed by the data sender. On input of the public parameters $prms$, the server's public key P_S, the receiver's ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$, and a keyword w, this algorithm outputs a searchable ciphertext C corresponding to w for the data receiver with ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$, which will be sent to the server.
- **Trapdoor**: This algorithm is performed by the data receiver or one of his ancestors. On input of the public parameters $prms$, the server's public key P_S, the receiver's ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$, the secret key of the receiver or his ancestor that runs this algorithm $s_{ID|_i}$ ($i \leq t$), and a keyword w, this algorithm outputs a trapdoor T corresponding to w, which will be sent to the server.
- **Test**: This algorithm is performed by the server. On input of the public parameters $prms$, the server's public key P_S and his secret key x_S, the receiver's ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$, the ciphertext C, and the trapdoor T, this algorithm outputs 1 if C and T correspond to the same keyword matches and 0 otherwise.

3.2 The Security Model

We now define our security model of a dHIBSE scheme. In dHIBSE schemes, the security of the searchable ciphertexts should be guaranteed. It means that we need to ensure that the output of the *dHIBSE* algorithm does not leak information about its corresponding keyword. More precisely, under an adaptive chosen keyword attack, it should be infeasible for every active PPT adversary to distinguish a ciphertext of a keyword w_0 from a ciphertext of a keyword w_1. Additionally, dHIBSE schemes should ensure the security against KGAs. It means that the adversary should not learn any information on w from a given trapdoor T_w. In [8], Rhee *et al.* proved that trapdoor-indistinguishability is a sufficient condition to ensure security against offline KGAs launched by outside adversaries. Therefore, under an adaptive chosen keyword attack, distinguishing a trapdoor of a keyword w_0 from a trapdoor of a keyword w_1 should be infeasible for every active PPT outside adversaries.

The security model of a dHIBSE scheme requires that:

(1) An inside adversary (the server) S who can ob-

tain trapdoors for any non-challenged keywords, should not be able to distinguish between the ciphertexts of two challenge keywords of its choice.

(2) An outside adversary (everyone except the server but including the receiver and his ancestors) $\mathcal{A}$ who can obtain trapdoors for any keywords, should not be able to distinguish between the ciphertexts of two challenge keywords of its choice.

(3) An outside adversary (everyone except the server, the receiver and his ancestors) $\mathcal{A}'$ who can obtain trapdoors for any keywords, should not be able to distinguish between the trapdoors of two chosen challenge keywords.

3.3 Ciphertext Security

Let $\mathcal{A}$ and S be the PPT adversaries described above. Suppose that $\Pi = (Setup, KeyGen, ServerKeyGen, dHIBSE, Trapdoor, Test)$ is a dHIBSE scheme. The following two games formally define ciphertext-indistinguishability for an outside adversary $\mathcal{A}$ and an inside adversary S, respectively.

Game I: Ciphertext-indistinguishability against $\mathcal{A}$: This game is performed between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

- *Initialization.* $\mathcal{B}$ generates a master secret key s, the public parameters $prms$, and the server's key pairs (x_S, P_S) by running $Setup(\lambda)$ and $ServerKeyGen$, respectively. Finally, it sends $prms$ and P_S to $\mathcal{A}$ and keeps s and x_S secret.
- *Phase 1.* The adversary $\mathcal{A}$ can adaptively make (a polynomially bounded number of) the following queries.
  - Queries to $ExtSK(ID|_t)$: When $\mathcal{A}$ supplies an ID-tuple $(ID|_t = (ID_1, ID_2, \cdots, ID_t))$ and requests his secret key, $\mathcal{B}$ returns the corresponding secret key $s_{ID|_t}$.
  - Queries to $Trapdoor(ID|_t, w)$: When $\mathcal{A}$ supplies a receiver with ID-tuple $(ID|_t = (ID_1, ID_2, \cdots, ID_t))$, and a keyword w, $\mathcal{B}$ computes a valid trapdoor and returns it to $\mathcal{A}$.
  - Queries to $Test(ID|_t, T, C)$: When $\mathcal{A}$ supplies a receiver with ID-tuple $(ID|_t = (ID_1, ID_2, \cdots, ID_t))$, a trapdoor T, and a searchable ciphertext C, $\mathcal{B}$ returns 1 if the keyword corresponding to T and C matches and 0 otherwise.
- *Challenge.* Eventually, $\mathcal{A}$ outputs a receiver's ID-tuple $ID^*|_t$ and two different keywords w_0 and w_1 as the challenge keywords. Then, $\mathcal{B}$ randomly selects a bit $b \in \{0,1\}$ and generates the searchable ciphertext C^* corresponding to w_b. Finally, it sends the challenge ciphertext C^* to $\mathcal{A}$.
- *Phase 2.* $\mathcal{A}$ can continue to make queries as in Phase 1.
- *Guess.* At last, $\mathcal{A}$ outputs $b' \in \{0,1\}$ as its guess. $\mathcal{A}$ wins the game if $b = b'$ and the following restriction holds:
  - $\mathcal{A}$ has never queried to $Test(ID^*|_t, T_{w_i}, C^*)$ for $i = 0, 1$.

The advantage of $\mathcal{A}$ is expressed as

$$Adv^{indC}_{\mathcal{A},\Pi}(\lambda) = |2Pr[b' = b] - 1|.$$

Game II: Ciphertext-indistinguishability against $\mathcal{S}$: This game is performed between an adversary $\mathcal{S}$ and a challenger $\mathcal{B}$.

- *Initialization.* $\mathcal{B}$ generates a master secret key s, the public parameters $prms$, and the server's key pairs (x_S, P_S) by running $Setup(\lambda)$ and $ServerKeyGen$, respectively. Finally, it sends $prms$ and x_S to $\mathcal{S}$ and keeps s secret.
- *Phase 1.* The adversary $\mathcal{S}$ can adaptively execute queries for a polynomially-bounded number of iterations as defined in Game I, except for the Queries to $Test$, and $\mathcal{B}$ responds to them in the same way.
- *Challenge.* Eventually, $\mathcal{S}$ outputs a receiver's ID-tuple $ID^*|_t$ and two different keywords w_0 and w_1 as the challenge keywords. Then, $\mathcal{B}$ randomly selects a bit $b \in \{0,1\}$ and generates the searchable ciphertext C^* corresponding to w_b. Finally, it sends the challenge ciphertext C^* to $\mathcal{S}$.
- *Phase 2.* $\mathcal{S}$ can continue to make queries as in Phase 1.
- *Guess.* At last, $\mathcal{S}$ outputs $b' \in \{0,1\}$ as its guess. $\mathcal{S}$ wins the game if $b = b'$ and the following restrictions hold:
  - $\mathcal{S}$ has never queried to $ExtSK$ on input of $(ID^*|_t)$ or one of his ancestors.
  - $\mathcal{S}$ has never queried to $Trapdoor(ID^*|_t, w_i)$ for $i = 0, 1$.

The advantage of $\mathcal{S}$ is expressed as

$$Adv^{indC}_{\mathcal{S},\Pi}(\lambda) = |2Pr[b' = b] - 1|.$$

Definition 4. A dHIBSE scheme Π satisfies ciphertext-indistinguishability under an adaptive chosen keyword attack if for any polynomially bounded adversary $\mathcal{X}$ $(\mathcal{X} \in \{\mathcal{S}, \mathcal{A}\})$, $Adv^{indC}_{\mathcal{X},\Pi}(\lambda)$ is negligible.

3.4 Trapdoor Security

Let $\mathcal{A}'$ be the PPT adversary described as above. Suppose that $\Pi = (Setup, KeyGen, ServerKeyGen, dHIBSE, Trapdoor, Test)$ is a dHIBSE scheme.

The following game formally defines trapdoor-indistinguishability for an outside adversary (everyone except the server and the receiver) $\mathcal{A}'$.

Game III: Trapdoor-indistinguishability against $\mathcal{A}'$: This game is performed between an adversary $\mathcal{A}'$ and a challenger $\mathcal{B}$.

- *Initialization.* $\mathcal{B}$ generates a master secret key s, the public parameters $prms$, and the server's key pairs (x_S, P_S) by running $Setup(\lambda)$ and $ServerKeyGen$, respectively. Finally, it sends $prms$ and P_S to $\mathcal{S}$ and keeps s and x_S secret.
- *Phase 1.* The adversary $\mathcal{A}'$ can adaptively execute queries for a polynomially-bounded number of iterations as defined in Game I and $\mathcal{B}$ responds to them in the same way.
- *Challenge.* Eventually, $\mathcal{A}'$ outputs a receiver's ID-tuple $ID^*|_t$ and two challenge keywords w_0 and w_1 $(\neq w_0)$. Then, $\mathcal{B}$ randomly selects a bit $b \in \{0,1\}$ and generates the trapdoor T^* corresponding to w_b. Finally, it sends T^* as the challenge trapdoor to $\mathcal{A}'$.
- *Phase 2.* $\mathcal{A}'$ can continue to make queries as in Phase 1.
- *Guess.* At last, $\mathcal{A}'$ outputs $b' \in \{0,1\}$ as its guess. $\mathcal{A}'$ wins the game if $b = b'$ and the following restrictions hold:
  - $\mathcal{A}'$ has never queried to on input of $(ID^*|_t)$ or one of his ancestors.
  - $\mathcal{A}'$ has never queried to $Test(ID^*|_t, T^*, *)$ where $*$ can be any arbitrary value in the corresponding domain.

The advantage of $\mathcal{A}'$ is expressed as

$$Adv^{indT}_{\mathcal{A}',\Pi}(\lambda) = |2Pr[b' = b] - 1|.$$

Definition 5. A dHIBSE scheme Π satisfies trapdoor-indistinguishability under an adaptive chosen keyword attack if for any polynomially bounded adversary $\mathcal{A}'$, $Adv^{indT}_{\mathcal{A}',\Pi}(\lambda)$ is negligible.

4 The Proposed dHIBSE Scheme

In this section, we describe the details of our proposed dHIBSE scheme.

- **Setup:** performed by the root PKG.
  - *Input:* The security parameter λ.
  - *Process:*
    1. Picks up two cyclic groups G_1 and G_2 of prime order $q > 2^\lambda$.
    2. Choose a generator $P \in G_1$.
    3. Choose a bilinear map $e : G_1 \times G_1 \rightarrow G_2$.
    4. Choose $s \in Z_q^*$ at random as the master secret key, and compute $P_{pub} = sP \in G_1$.
    5. Choose cryptographic hash functions: $h : G_1 \times \{0,1\}^* \rightarrow Z_q^*$, $H_1 : \{0,1\}^* \rightarrow G_1$, $h_2 : G_1 \times G_1 \times \{0,1\}^* \times G_1 \times G_2 \rightarrow Z_q^*$, $h_3 : \{0,1\}^* \rightarrow Z_q^*$, $H_4 : G_1 \times \{0,1\}^* \rightarrow G_1$, $h_5 : G_1 \times G_1 \times \{0,1\}^* \times G_1 \rightarrow Z_q^*$ and $h_6 : Z_q^* \times \{0,1\}^* \rightarrow Z_q^*$.
  - *Output:* The public parameters $prms = (G_1, G_2, e, q, P, P_{pub}, h, H_1, h_2, h_3, H_4, h_5, h_6)$ and the master secret key s.
- **KeyGen:** performed by the root PKG or a lower-level PKG.
  - *Input:* Public parameters $prms$, ID-tuple $ID|_{t+1} = (ID_1, \cdots, ID_{t+1})$ and the secret key of the ancestor of $ID|_{t+1}$, i.e., $s_{ID|_t}$ (note that if $t = 1$, then $s_{ID|_t} = s$).
  - *Process:*
    1. Compute $r_{ID|_{t+1}} = h_6(s_{ID|_t}, ID|_{t+1})$.
    2. Compute $R_{ID|_{t+1}} = r_{ID|_{t+1}} P$.
    3. Compute $s_{ID|_{t+1}} = r_{ID|_{t+1}} + s_{ID|_t} h(R_{ID|_{t+1}}, ID|_{t+1})$.
  - *Output:* $(s_{ID|_{t+1}}, R_{ID|_{t+1}})$.
- **ServerKeyGen:** performed by the server.
  - *Input:* Public parameters $prms$.
  - *Process:* Choose a random number $x_S \in Z_q^*$ and compute $P_S = x_S P$.
  - *Output:* x_S and P_S.
- **dHIBSE:** performed by the data sender.
  - *Input:* Public parameters $prms$, the server's public key P_S, the receiver's ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$, $R_{ID|_1}$, $R_{ID|_2}$, $\cdots$, $R_{ID|_t}$ and a keyword w.
  - *Process*: Choose a random number $r \in Z_q^*$ and compute

$$C_1 = rP$$

$$C_2 = rH_4\left(R_{ID|_t}, ID|_t\right)$$

$$\begin{aligned} K = & \sum_{i=1}^{t} \prod_{j=i+1}^{t} h\left(R_{ID|_j}, ID|_j\right) R_{ID|_i} \\ & + \prod_{i=1}^{t} h\left(R_{ID|_i}, ID|_i\right) P_{pub} \\ & = \left[h\left(R_{ID|_2}, ID|_2\right) \cdots h\left(R_{ID|_t}, ID|_t\right)\right] R_{ID|_1} \\ & + \left[h\left(R_{ID|_3}, ID|_3\right) \cdots h\left(R_{ID|_t}, ID|_t\right)\right] R_{ID|_2} \\ & + \cdots + R_{ID|_t} + h\left(R_{ID|_1}, ID|_1\right) \cdot h(R_{ID|_2} \\ & , ID|_2) \cdot h\left(R_{ID|_2}, ID|_2\right) \cdot h\left(R_{ID|_3}, ID|_3\right) \ldots \\ & \cdot h\left(R_{ID|_t}, ID|_t\right) P_{pub} \end{aligned}$$

$$C_3 = h_2\left[C_1, P_S, ID|_t, r \cdot P_S, e\left(rH_1(w), K\right)\right]$$

  - *Output*: The searchable ciphertext $C = (C_1, C_2, C_3)$.
- **Trapdoor:** performed by the data receiver or one of his ancestors.

- *Input:* Public parameters $prms$, the server's public key P_S, a keyword w, the receiver's ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$ and the secret key of the receiver or his ancestor that runs this algorithm $s_{ID|_i}$.
- *Process:*
  1. (1) If the receiver is not the one that performs this algorithm, then use $s_{ID|_i}$ step by step down and compute $s_{ID|_t}$ as specified in the KeyGen algorithm.
  2. (2) Choose random numbers $r_1, r_2 \in Z_q^*$ and compute

$$T_1 = r_1 P$$
$$T_2 = r_2 h_3(w) h_5 (T_1, P_S, ID|_t, r_1 P_S) P$$
$$T_3 = s_{ID|_t} H_1(w) + r_2 h_3(w) H_4 \left(R_{ID|_t}, ID|_t\right)$$

- *Output:* The trapdoor $T = (T_1, T_2, T_3)$.

- **Test:** performed by the server.
  - *Input:* Public parameters $prms$, the server's public key P_S and his secret key x_S, the receiver's ID-tuple $ID|_t = (ID_1, \cdots, ID_t)$, the trapdoor $T = (T_1, T_2, T_3)$, and the ciphertext $C = (C_1, C_2, C_3)$.
  - *Process:*
    - Compute $h' = h_5(T_1, P_S, ID|_t, x_S T_1)$ and check if the following equation holds:

$$C_3 = h_2 \left(C_1, P_S, ID|_t, x_S C_1, \frac{e(T_3, C_1)}{e(T_2, C_2)^{h'^{-1}}}\right)$$

  - *Output:* 1 if the above equation holds and 0 otherwise.

In the following, we first show that the proposed scheme works correctly.

Theorem 1. *Let $C = (C_1, C_2, C_3)$ and $T = (T_1, T_2, T_3)$ be the ciphertext and the trapdoor corresponding to the keywords w and w'. Then, if $w = w'$, the output of the Test algorithm on C and T would be 1.*

Proof. Suppose that $w = w'$. Now, we will prove the correctness of the proposed dHIBSE scheme as follows:

$$\begin{aligned} C_3 &= h_2 \left(C_1, P_S, ID|_t, rP_S, e(rH_1(w), K)\right) \\ &= h_2(C_1, P_S, ID|_t, rP_S, e(s_{ID|_t} H_1(w) \\ &+ r_2 h_3(w) H_4(R_{ID|_t}, ID|_t), rP) e(r_2 h_3(w_i) h_5(T_w^1, \\ &P_S, ID|_t, r_1 P_S) P, rH_4(R_{ID|_t}, ID|_t))^{-(h'^{-1})}) \end{aligned} \quad (1)$$

$$\begin{aligned} &= h_2 \left(C_1, P_S, ID|_t, rP_S, \frac{e(T_w^3, C_1)}{e(T_w^2, C_2)^{h'^{-1}}}\right) \\ &= h_2(C_1, P_S, ID|_t, rx_S P, \frac{e(T_w^3, C_1)}{e(T_w^2, C_2)^{h'^{-1}}}) \\ &= h_2(C_1, P_S, ID|_t, x_S C_1, \frac{e(T_w^3, C_1)}{e(T_w^2, C_2)^{h'^{-1}}}) \end{aligned}$$

5 Security Analysis

In this section, we prove that our proposed dHIBSE scheme meets the security requirements. Let Π=(Setup, KeyGen, ServerKeyGen, dHIBSE, Trapdoor, Test) be the dHIBSE scheme of Section 4. In the following, we provide Theorem 3 and Theorem 4 to show that Π satisfies indistinguishability of ciphertexts and trapdoors in the sense of Definition 4 and Definition 5, respectively.

Theorem 2. *Under the hardness assumption of CDH and GBDH problems, the proposed dHIBSE scheme of Section 4 satisfies ciphertext-indistinguishability under an adaptive chosen keyword attack.*

Proof. We defer the proof of this theorem to the full version of the paper.

Theorem 3. *Under the hardness assumption of CDH and GBDH problems, the proposed dHIBEKS scheme of Section 4 satisfies ciphertext-indistinguishability under an adaptive chosen keyword attack.*

Proof. In order to prove this theorem, we should show that for any polynomially bounded adversaries $\mathcal{A}$ and $\mathcal{S}$, $Adv_{\mathcal{A},\Pi}^{indC}(\lambda)$ and $Adv_{\mathcal{S},\Pi}^{indC}(\lambda)$ are negligible. This theorem will be proved through Lemma 1 and Lemma 2.

Lemma 1. *For every inside adversary $\mathcal{A}$, $Adv_{\mathcal{A},\Pi}^{indC}(\lambda)$ is negligible.*

Proof. Suppose that there exists a polynomially bounded outside adversary (including the receiver and his ancestors) $\mathcal{A}$ against the proposed scheme that wins Game I with a non-negligible advantage ϵ (i.e., $Adv_{\mathcal{A},\Pi}^{indC}(\lambda) = \epsilon$). Then, we can use $\mathcal{A}$ to build a polynomially bounded algorithm $\mathcal{B}$ to solve CDH problem with the same advantage. Therefore, based on the hardness assumption of the CDH problem, we conclude that ϵ should be negligible.

Given a random instance of the CDH problem (P, xP, yP), we show how $\mathcal{B}$ can simulate the challenger of Game I and find the answer to the problem.

- *Initialization.* $\mathcal{B}$ runs the Setup algorithm as specified in the proposed scheme except that it sets $P_S = xP$. It returns $prms$ and P_S to $\mathcal{A}$.
- *Phase 1.* $\mathcal{B}$ replies to $\mathcal{A}$'s queries. The list of queries that $\mathcal{A}$ can perform and the way $\mathcal{B}$

answers them is described in the following. In order to respond consistently to $\mathcal{A}$'s queries, $\mathcal{B}$ should maintain the following lists that are initially empty: L, L_1, L_2, L_3, L_4, L_5, L_6, and L_{key} (this assumption holds throughout the rest of the paper).

- Queries to h: On input $(R_{ID|_i}, ID|_i)$ to this oracle, $\mathcal{B}$ searches L to find the entry $(ID|_i, R_{ID|_i}, h)$ and returns h if such an entry is found. Otherwise, $\mathcal{B}$ chooses $h \in Z_q^*$ randomly and inserts $(ID|_1, R_{ID|_1}, h)$ to L and returns h to $\mathcal{A}$.
- Queries to H_1: On input w to this oracle, $\mathcal{B}$ searches L_1 for the entry (w, H_1) and returns H_1 if such an entry is found. Otherwise, it chooses $H_1 \in G_1$ randomly, inserts (w, H_1) to L_1 and returns H_1 to $\mathcal{A}$.
- Queries to h_2: On input $(C_1, P_S, ID|_t, P', g)$ to this oracle, if $e(xP, yP) = e(P', P)$, then $\mathcal{B}$ returns P' as the answer to the instance of CDH problem and stops. Otherwise, it searches L_2 for the entry $(C_1, P_S, ID|_t, P', g, h_2)$ and returns h_2 in case such an entry is found. Otherwise, it chooses $h_2 \in Z_q^*$ randomly, inserts $(C_1, P_S, ID|_t, P', g, h_2)$ to L_2 and returns h_2.
- Queries to h_3: On input w to this oracle, $\mathcal{B}$ searches L_3 for the entry (w, h_3) and returns h_3 if such an entry is found. Otherwise, $\mathcal{B}$ chooses $h_3 \in Z_q^*$ randomly and inserts (w, h_3) to L_3 and returns h_3 to $\mathcal{A}$.
- Queries to H_4: On input $(R_{ID|_t}, ID|_t)$ to this oracle, $\mathcal{B}$ searches L_4 for the entry $(R_{ID|_t}, ID|_t, h_4, H_4)$ and returns H_4 if such an entry is found. Otherwise, $\mathcal{B}$ chooses $h_4 \in Z_q^*$ randomly, computes $H_4 = h_4 P$ and inserts $(R_{ID|_t}, ID|_t, h_4, H_4)$ to L_4. Then, it returns H_4 to $\mathcal{A}$.
- Queries to h_5: On input $(T_1, P_S, ID|_t, x_S T_1)$ to this oracle, $\mathcal{B}$ searches L_5 for the entry $(T_1, P_S, ID|_t, x_S T_1, h_5)$ and returns h_5 if such an entry is found. Otherwise, $\mathcal{B}$ chooses $h_5 \in Z_q^*$ randomly and inserts $(T_1, P_S, ID|_t, x_S T_1, h_5)$ to L_5 and returns h_5 to $\mathcal{A}$.
- Queries to h_6: On input $(s_{ID|_{t-1}}, ID|_t)$, $\mathcal{B}$ searches L_6 for the entry $(s_{ID|_{t-1}}, ID|_t, h_6)$ and returns h_6 if such an entry is found. Otherwise, $\mathcal{B}$ chooses $h_6 \in Z_q^*$ randomly and inserts $(s_{ID|_{t-1}}, ID|_t, h_6)$ to L_6 and returns h_6 to $\mathcal{A}$.
- Queries to ExtSK: On input $(ID|_t = (ID_1, ID_2, \cdots, ID_t))$ to this oracle, $\mathcal{B}$ searches L_{key} to find the tuple $(ID'|_i = (ID'_1, \cdots, ID'_i), R_{ID'|_i}, s_{ID'|_i})$ with maximum value of i such that for $j = 1 \cdots, i$: $ID_j = ID'_j$.
  - If $i = t$, $\mathcal{B}$ returns $(R_{ID|_t}, s_{ID|_t})$.
  - If no such an i is found, $\mathcal{B}$
    (1) Obtains $(s, ID|_1, h_6)$ by calling h_6 on $(s, ID|_1)$ and computes $R_{ID|_1} = h_6 P$.
    (2) Obtains $(R_{ID|_1}, ID|_1, h)$ by calling h on $(R_{ID|_1}, ID|_1)$ and computes $s_{ID|_1} = h_6 + sh$.
    (3) Adds $(ID|_1, R_{ID|_1}, s_{ID|_1})$ to L_{key} and sets $i = 1$.
  - If $i < t$, then for $j = i+1, \cdots, t$:
    (1) Obtains $(s_{ID|_{j-1}}, ID|_j, h_6)$ by calling h_6 on $(s_{ID|_{j-1}}, ID|_j)$ and computes $R_{ID|_j} = h_6 P$.
    (2) Obtains $(R_{ID|_j}, ID|_j, h)$ by calling h on $(R_{ID|_j}, ID|_j)$ and computes $s_{ID|_j} = h_6 + s_{ID|_{j-1}} h$.
    (3) Adds tuple $(ID|_j, R_{ID|_j}, s_{ID|_j})$ to L_{key} and returns the tuple $(R_{ID|_j}, s_{ID|_j})$ if $j = t$.
- Queries to Trapdoor: On input $(ID|_t, w)$ to this oracle, $\mathcal{B}$
  (1) Obtains $(ID|_t, R_{ID|_t}, s_{ID|_t})$, $(R_{ID|_t}, ID|_t, h_4, H_4)$, (w, H_1) and (w, h_3) by calling $ExtSK$, H_4, H_1 and h_3 on $ID|_t$, $(R_{ID|_t}, ID|_t)$, w and w, respectively.
  (2) Chooses $r_1, r_2 \in Z_q^*$ randomly and computes

$$\begin{aligned} T_1 &= r_1 P \\ T_2 &= r_2 h_3(w) h_5\left(T_1, P_S, ID|_t, r_1 P_S\right) P \\ T_3 &= s_{ID|_t} H_1(w) + r_2 h_3(w) H_4\left(R_{ID|_t}, ID|_t\right). \end{aligned}$$

  (3) Returns $T = (T_1, T_2, T_3)$.
- Queries to Test: On input $(ID|_t$, $C = (C_1$, C_2, $C_3)$, $T = (T_1, T_2, T_3))$ to this oracle, $\mathcal{B}$
  (1) Searches L_5 for some tuple $(T_1, P_S, ID|_t, *, h_5)$. If no such a tuple is found, it chooses $h_5 \in Z_q^*$ at random and inserts $(T_1, P_S, ID|_t, \perp, h_5)$ to L_5.
  (2) Searches L_2 for some tuple $(C_1, P_S, ID|_t, *, \dfrac{e(T_3, C_3)}{e(T_2, C_2)^{h_5^{-1}}}, h_2)$. If no such a tuple is found, $\mathcal{B}$ chooses $h'_2 \in Z_q^*$ randomly and inserts

$$(C_1, P_S, ID|_t, \perp, \frac{e(T_3, C_3)}{e(T_2, C_2)^{h_5^{-1}}}, h'_2)$$

  to L_2.
  (3) Outputs 1 if $C_3 = h'_2$ and 0 otherwise.

- *Challenge.* Eventually, $\mathcal{A}$ outputs a receiver's ID-tuple $ID^C|_t$ and two different keywords w_0 and w_1 as the challenge keywords. Then, $\mathcal{B}$ randomly selects a bit $b \in \{0,1\}$ and obtains $(ID^C|_t, R_{ID^C|_t}, s_{ID^C|_t})$, $(R_{ID^C|_t}, ID^C|_t, h_4, H_4)$ and (w_b, H_1^b) by calling $ExtSK$, H_4 and H_1 on $(ID^C|_t)$, $(R_{ID^C|_t}, ID^C|_t)$ and w_b, respectively. Afterwards, it sets $C_1 = yP$, $C_2 = h_4yP$ and $C_3 = h_2$ for a random value $h_2 \in Z_q^*$, and adds the tuple $(C_1, P_S, ID^C|_t, \perp, e(H_1^b, s_{ID^C|_t}yP), h_2)$ to L_2. Finally, it sends (C_1, C_2, C_3) as the challenge ciphertext to $\mathcal{A}$.
- *Phase 2.* $\mathcal{A}$ can continue to make queries as in Phase 1.
- *Guess.* At last, $\mathcal{A}$ outputs $b' \in \{0,1\}$ as its guess.

Now, we proceed to compute the advantage of $\mathcal{B}$ in solving CDH. In this order, considering that h_2 is modeled as a random oracle, it can be deduced that $\mathcal{A}$'s advantage would be negligible unless $(C_1, P_S, ID|_t, xyP, e(yH_1(w), K))$ appears on L_2. If it appears on L_2, then $\mathcal{B}$ certainly is able to solve the CDH problem. Consequently, we can conclude that the advantage of $\mathcal{B}$ is equal to ϵ. □

Lemma 2. *For every inside adversary $\mathcal{S}$, $Adv_{\mathcal{S},\Pi}^{indC}(\lambda)$ is negligible.*

Proof. Suppose that there exists an inside adversary $\mathcal{S}$ against the proposed scheme that wins Game II with a non-negligible advantage ϵ (i.e., $Adv_{\mathcal{S},\Pi}^{indC}(\lambda) = \epsilon$). Then, we can use $\mathcal{S}$ to build a polynomially bounded algorithm $\mathcal{B}$ to solve $GBDH$ problem with an advantage at least equal to $\dfrac{\epsilon l}{eq_h(q_{H_1}+1)}$ (for polynomially bounded integers q_h, q_{H_1}, l). Therefore, based on the hardness assumption of the $GBDH$ problem, we conclude that ϵ should be negligible.

Given a random instance of the $GBDH$ problem (P, xP, yP, zP), we show how $\mathcal{B}$ can simulate the challenger of Game II and find the answer to the problem.

- *Initialization.* $\mathcal{B}$ runs the Setup algorithm as it is specified in the proposed scheme except that it sets $P_{pub} = xP$. It also chooses $x_S \in Z_q^*$ randomly and sets $P_S = x_SP$. Then, it chooses some values $l_i \leq q$ randomly for $1 \leq i \leq l$ as the indices corresponding to the challenge ID-tuple of data receiver, where l denotes the maximum hierarchical level and q represents the maximum number of ID-tuples with the same parent. Let $q_h = q^l$ denotes the maximum number of queries that $\mathcal{S}$ could query from the H oracle on different ID-tuples. Finally, it returns $(prms, x_S, P_S)$ to $\mathcal{S}$.
- *Phase 1.* $\mathcal{B}$ replies to queries of $\mathcal{S}$. The queries that can be made by $\mathcal{S}$ and $\mathcal{B}$'s answer to them are the same as those in Lemma 1 except for the following queries.
  - Queries to H_1: On input w to this oracle, $\mathcal{B}$ searches L_1 for the entry $(w, coin, \mu, H_1)$ and returns H_1 if such an entry is found. Otherwise, it generates $coin \in \{0,1\}$ such that $Pr[coin = 0] = \frac{1}{q_{H_1}+1}$. Then, it chooses $\mu \in Z_q^*$ at random, computes $H_1 = (1-coin)yP + \mu P$, adds $(w, coin, \mu, H_1)$ to L_1 and returns H_1 to $\mathcal{S}$.
  - Queries to h_2: On input $(C_1, P_S, ID|_t, P', g)$ to this oracle, $\mathcal{B}$
    (1) For each keyword w queried so far to H_1:
      - Calls H_1 on w and obtains $(w, coin, \mu, H_1)$.
      - Checks if the output of $DBDH$ oracle On input $(xP, yP, zP, \dfrac{g}{e(xP,zP)^\mu})$ is equal to 1 or not. If it is 1, returns $\dfrac{g}{e(xP,zP)^\mu}$ as the answer to the instance of $GBDH$ problem and stops.

    (2) Searches L_2 for the tuple $(C_1, P_S, ID|_t, P', g, h_2)$. If no such entry is found and $e(C_1, P_S) = e(P, P')$, it chooses $h_2 \in Z_q^*$ randomly and adds $(C_1, P_S, ID|_t, P', g, h_2)$ to L_2. In both cases, it returns h_2.
  - Queries to ExtSK: On input $(ID|_t{=}(ID_1, ID_2, \cdots, ID_t))$ to this oracle, $\mathcal{B}$ searches L_{key} to find the tuple $(n_1, n_2, \cdots, n_i, 0, \cdots, 0, ID'|_i{=}\ (ID'_1, \cdots, ID'_i), R_{ID'|_i}, s_{ID'|_i})$ with maximum value of i such that for $j = 1\cdots, i$: $ID_j = ID'_j$. If $i = t$, $\mathcal{B}$ returns $s_{ID|_t}$. Otherwise, it proceeds as follows:
    (1) If no such i is found,
      (a) Searches L_{key} to find the entry $(n_1, 0, \cdots, 0, ID'|_1, R_{ID'|_1}, s_{ID'|_1})$ by maximum value of n_1.
        - If $n_1 = l_1 - 1$, it chooses $h \in Z_q^*$ randomly, computes $R_{ID|_1} = xP - hxP$ and adds the tuples $(n_1 + 1, 0, \cdots, 0, ID|_1, R_{ID|_1}, s_{ID|_1})$, $(ID|_1, R_{ID|_1}, h)$ and $(\perp, ID|_1, \perp)$ to L_{key}, L and L_6, respectively.
        - Otherwise, it chooses $h, s_{ID|_1} \in Z_q^*$ randomly, computes $R_{ID|_1} =$

$s_{ID|_1}P - hxP$ and adds tuples $(n_1+1,\ 0,\ \cdots,0,\ ID|_1,\ R_{ID|_1},\ s_{ID|_1})$, $(ID|_1, R_{ID|_1}, h)$ and $(\perp, ID|_1, \perp)$ to L_{key}, L and L_6, respectively.

(b) Sets $i = 1$.

(2) Sets $i = i + 1$.

(3) Searches L_{key} to find the entry $(n_1, n_2, \cdots, n_i, 0, \cdots, 0, ID'|_i{=}(ID_1, \cdots, ID_{i-1}, ID'_i), R_{ID'|_i}, s_{ID'|_i})$ by maximum value of n_i. If no such a tuple is found, searches L_{key} to find the entry $(n_1, n_2, \cdots, n_{i-1}, 0, \cdots, 0, ID'|_{i-1}{=} (ID_1, \cdots, ID_{i-1}), R_{ID|_{i-1}}, s_{ID|_{i-1}})$. In both cases, it proceeds as follows:

- If $n_1 = l_1, n_2 = l_2, \cdots, n_i = l_i - 1$, it chooses $h \in Z_q^*$ randomly, computes $R_{ID|_i} = xP - hxP$ and adds tuples $(n_1, \cdots, n_i{+}1, \cdots, 0, ID|_i, R_{ID|_i}, \perp)$, $(ID|_i, R_{ID|_i}, h)$ and $(\perp, ID|_i, \perp)$ to L_{key}, L and L_6, respectively.
- If $n_1 = l_1, n_2 = l_2, \cdots, n_i \neq l_i - 1$, it chooses $h, s_{ID|_i} \in Z_q^*$ randomly, computes $R_{ID|_i} = s_{ID|_i}P - hxP$ and adds tuples $(n_1, \cdots, n_i + 1, \cdots, 0, ID|_i, R_{ID|_i}, s_{ID|_i})$, $(ID|_i, R_{ID|_i}, h)$ and $(\perp, ID|_i, \perp)$ to L_{key}, L and L_6, respectively.
- If one of the inequalities $n_1 \neq l_1, n_2 \neq l_2, \cdots, n_{i-1} \neq l_{i-1}$ holds, it obtains $(s_{ID|_{i-1}}, ID|_i, h_6)$ by calling h_6 on $(s_{ID|_{i-1}},\ ID|_i)$ and computes $R_{ID|_i} = h_6 P$. Then, it obtains $(ID|_i,\ R_{ID|_i}, h)$ by calling h on $(ID|_i,\ R_{ID|_i})$ and sets $s_{ID|_i} = h_6 +\ s_{ID|_{i-1}}h$. Finally, it adds tuple $(ID|_i, R_{ID|_i}, s_{ID|_i})$ to L_{key}, respectively.

(4) Goes to the step 2 if $i \neq t$, .

(5) Return $s_{ID|_t}$ if $s_{ID|_t} \neq \perp$, and aborts otherwise.

◦ Queries to Trapdoor: On input $(ID|_t, w)$ to this oracle, $\mathcal{B}$ obtains $(ID|_t, R_{ID|_t}, s_{ID|_t})$, $(R_{ID|_t}, ID|_t, h_4, H_4)$, $(w, coin, \mu, H_1)$ and (w, h_3) by calling $ExtSK$, H_4, H_1 and h_3 on $ID|_t$, $(R_{ID|_t}, ID|_t)$, w and w, respectively. Then, it selects $r_1, r_2 \in Z_q^*$ randomly and proceeds as follows:

- If $s_{ID|_t} \neq \perp$, $\mathcal{B}$ computes

$$\begin{aligned} T_1 &= r_1 P \\ T_2 &= r_2 h_3(w) h_5(T_1, P_S, ID|_t, r_1 P_S) P \\ T_3 &= s_{ID|_t} H_1(w) + r_2 h_3(w) H_4(R_{ID|_t}, ID|_t). \end{aligned}$$

- If $s_{ID|_t} = \perp$ and $coin = 0$, $\mathcal{B}$ aborts.
- If $s_{ID|_t} = \perp$ and $coin = 1$, it computes

$$\begin{aligned} T_1 &= r_1 P \\ T_2 &= r_2 h_3(w) h_5(T_1, P_S, ID|_t, r_1 P_S) P \\ T_3 &= \mu x P + r_2 h_3(w) H_4(R_{ID|_t}, ID|_t). \end{aligned}$$

- Returns $T = (T_1, T_2, T_3)$.

◦ Queries to Test: On input $(ID|_t,\ C = (C_1,\ C_2,\ C_3),\ T = (T_1,\ T_2,\ T_3))$ to this oracle, $\mathcal{B}$ obtains h_5 by calling h_5 On input $(T_1, P_S, ID|_t, x_S T_1, h_5)$. Then, it obtains h_2 by calling h_2 On input $(C_1, P_S, ID|_t, x_S C_1, \dfrac{e(T_3, C_3)}{e(T_2, C_2)^{h_5^{-1}}})$. Finally, $\mathcal{B}$ outputs 1 if $C_3 = h'_2$ and 0 otherwise.

- *Challenge.* Eventually, $\mathcal{S}$ outputs a receiver's ID-tuple $ID^C|_t{=}\ (ID_1^C, ID_2^C, \cdots, ID_t^C)$ and two different keywords w_0 and w_1 as the challenge keywords. $\mathcal{B}$ obtains $(w_0, coin_0, \mu_0, H_1^0)$ and $(w_1, coin_1, \mu_1, H_1^1)$ by calling H_1 on w_0 and w_1, respectively. It aborts if $coin_0 = coin_1 = 1$. Otherwise, it randomly selects a bit $b \in \{0, 1\}$ and obtains $(ID^C|_t, R_{ID^C|_t}, s_{ID^C|_t})$, $(R_{ID^C|_t}, ID^C|_t, h_4, H_4)$ and (w_b, h_3^b) by calling $ExtSK$, H_4 and h_3 on $ID^C|_t$, $(R_{ID^C|_t}, ID^C|_t)$ and w_b, respectively. $\mathcal{B}$ aborts if $s_{ID^C|_t} \neq \perp$. Otherwise, it sets $C_1 = zP$, $C_2 = h_4 zP$ and $C_3 = h_2$ for a random value $h_2 \in Z_q^*$ and inserts the tuple $(C_1, P_S, ID^C|_t, x_S zP, \perp, h_2)$ to L_2. Finally, it sends (C_1, C_2, C_3) as the challenge ciphertext to $\mathcal{S}$.
- *Phase 2.* $\mathcal{S}$ can continue to make queries as in Phase 1.
- *Guess.* At last, $\mathcal{S}$ outputs $b' \in \{0, 1\}$ as its guess.

We now define three events as follows:

- E_1: $\mathcal{B}$ does not abort as a result of any of $\mathcal{S}$'s trapdoor queries.
- E_2: $\mathcal{B}$ does not abort during the challenge phase.

$$Pr[E_1] \geq \left(1 - \frac{1}{q_{H_1} + 1}\right)^{q_{H_1}} \geq \frac{1}{e}$$

Note that $l_i \leq q$ $(for 1 \leq i \leq l)$ are independent of the adversary's view and L contains at most q_h elements. consequently, $\mathcal{B}$ correctly guesses a challenge ID-tuple $ID^C|_t{=}\ (ID_1^C, \ldots, ID_t^C)$, with probability $\frac{l}{q_h}$. Moreover, in the challenge phase, $\mathcal{B}$ aborts if both $coin_0 = 1$ and $coin_1 = 1$. Therefore, $Pr[E_2] = \left(\frac{l}{q_h}\right)\left(1 - \left(1 - \frac{1}{q_{H_1}+1}\right)^2\right) \geq \left(\frac{l}{q_h}\right)\left(\frac{1}{q_{H_1} + 1}\right)$.

Since the two events E_1 and E_2 are independent, $Pr[E_1 \wedge E_2] \geq \left(\frac{l}{q_h}\right)\left(\frac{1}{e(q_{H_1}+1)}\right)$. In this order, considering that h_2 is modeled as a random oracle, it can be deduced that the adversary's advantage would be negligible unless $((C_1, P_S, ID|_t, P', g), h_2)$ appears on L_2. Following this observation and the fact that $\mathcal{B}$ has made at most q_h queries from O_{DBDH}, we can conclude that $\epsilon' \geq \epsilon\left(\frac{l}{q_h}\right)\left(\frac{1}{e(q_{H_1}+1)}\right) = \frac{\epsilon l}{e q_h (q_{H_1}+1)}$ (the same as ϵ) is non-negligible. □

Theorem 4. *Under the hardness assumption of CDH problem, the proposed dHIBEKS scheme of Section 4 satisfies trapdoor-indistinguishability under an adaptive chosen keyword attack.*

Proof. According to the Definition 5, To prove this theorem, we should show that for any polynomially bounded adversary $\mathcal{A}'$, $Adv^{indT}_{\mathcal{A}',\Pi}(\lambda)$ is negligible.

Suppose that there exists a polynomially bounded outside adversary $\mathcal{A}'$ against the proposed scheme that wins Game III with a non-negligible advantage ϵ (i.e., $Adv^{indT}_{\mathcal{A}',\Pi}(\lambda) = \epsilon$). Then, we can use $\mathcal{A}'$ to build a polynomially bounded algorithm $\mathcal{B}$ to solve CDH problem with the same advantage. Therefore, based on the hardness assumption of solving CDH problem, it can be concluded that ϵ should be negligible.

Given a random instance of the CDH problem (P, xP, yP), we show how $\mathcal{B}$ can simulate the challenger of Game III and find the answer to the problem.

- *Initialization.* $\mathcal{B}$ runs the Setup algorithm as specified in the proposed scheme and sets $P_S = xP$. Then, it returns $prms$ and P_S to $\mathcal{A}'$.
- *Phase 1.* $\mathcal{B}$ replies to $\mathcal{A}'$'s queries. The queries that can be maid by $\mathcal{A}'$ and $\mathcal{B}$'s answer to them are the same as those in Lemma 1 except for the following queries.
  - Queries to h_2: On input $(C_1, P_S, ID|_t, P', g)$ to this oracle, $\mathcal{B}$ searches L_2 for the entry $(C_1, P_S, ID|_t, P', g, h_2)$ and returns h_2 in case such an entry is found. Otherwise, it selects $h_2 \in Z_q^*$ at random, inserts $(C_1, P_S, ID|_t, P', g, h_2)$ to L_2 and returns h_2.
  - Queries to h_5: On input $(T_1, P_S, ID|_t, P')$ to this oracle, $\mathcal{B}$ checks if $e(xP, yP) = e(P', P)$. It returns P' as the answer to the CDH problem if the equation holds and terminates. Otherwise, it searches L_5 for the entry $(T_1, P_S, ID|_t, P', h_5)$. If no such entry is found and $e(T_1, P_S) = e(P', P)$, it selects $h_5 \in Z_q^*$ at random and inserts $(T_1, P_S, ID|_t, P', h_5)$ to L_5. In both cases, it returns h_5.
- *Challenge.* Eventually, $\mathcal{A}'$ outputs a receiver's ID-tuple $ID^C|_t = (ID_1^C, ID_2^C, \cdots, ID_t^C)$ and two different keywords w_0 and w_1 as the challenge keywords. Then, $\mathcal{B}$ randomly selects a bit $b \in \{0,1\}$ and obtains (w_b, H_1^b), $(ID^C|_t, R_{ID^C|_t}, s_{ID^C|_t})$, $(R_{ID^C|_t}, ID^C|_t, h_4, H_4)$ and (w_b, h_3^b) by calling H_1, $ExtSK$, H_4 and h_3 On input w_b, $ID^C|_t$, $(R_{ID^C|_t}, ID^C|_t)$ and w_b, respectively. Then, it chooses $r_2 \in Z_q^*$ randomly and sets $T_1 = yP$, $T_2 = r_2 h_3^b h_5 H_4$ and $T_3 = s_{ID^C|_t} H_1^b + r_2 h_3^b$ for a random value $h_5 \in Z_q^*$. Finally, it inserts the tuple $(T_1, P_S, ID^C|_t, \perp, h_5)$ to L_5 and sends (T_1, T_2, T_3) as the challenge trapdoor to $\mathcal{A}'$.
- *Phase 2.* $\mathcal{A}'$ can continue to make queries as in Phase 1.
- *Guess.* At last, $\mathcal{A}'$ outputs $b' \in \{0,1\}$ as its guess.

Now, we proceed to compute the advantage of $\mathcal{B}$ in solving CDH. In this order, considering that h_5 is modeled as a random oracle, it can be deduced that $\mathcal{A}'$'s advantage would be negligible unless $(T_1, P_S, ID^C|_t, abP, h_5)$ appears on L_5. If it appears on L_5, then $\mathcal{B}$ certainly is able to solve the CDH problem. Consequently, we can conclude that the advantage of $\mathcal{B}$ is equal to ϵ. □

6 Performance Analysis

In this section, we compare our proposed scheme with Wang *et al.*'s scheme [3], Li *et al.*'s scheme [4], Liu *et al.*'s scheme [5], and some other related dIBEKS schemes [9, 21] in terms of both their computational and communication complexities, and their provided security requirements. The comparison in terms of provided security requirements and functionality are presented in Table 2. From Table 2, we observe that our proposed scheme and the schemes of [3–5] are the only schemes that support hierarchical keyword search capability. However, Wang *et al.*'s scheme [3] suffers from the vulnerability caused by the offline KGAs launched by outside adversaries, and the dHPEKS schemes of [4, 5] suffer from complicated certificate management problem.

The comparison in terms of computational cost is shown in Table 1. For simplicity, we only calculate the number of time-consuming operations. The notations used in the performance comparison are defined as below:

(1) T_{bp}: Time cost for computing a bilinear pairing operation.
(2) T_H: Time cost for computing a Hash-to-point mapping operation.
(3) T_{sm}: Time cost for computing a scalar multipli-

cation operation.

We use running times reported for the above-mentioned operations in [41] performed on a laptop running Ubuntu 14.04 LTS with Intel(R) CoreTM i3-2310M CPU@2.10 GHz and 4 GB RAM memory by using PBC (pairing-based cryptography) library [42]: $T_H = 0.334$ ms, $T_{sm} = 0.311$ ms, $T_{bp} = 2.486$ ms where ms means millisecond. According to Table 1 and Figure 3, the running times of the Key generation algorithm of our proposed scheme has fallen by 80.57% and 76.31%, respectively, compared to [4] and [5]. Moreover, the running times of the Ciphertext generation algorithm of our proposed scheme is significantly reduced by 44.84% and 48.38%, respectively, compared to [4] and [5] (For the sake of simplicity, assuming the hierarchical level of the data receiver $t = 3$, and the maximum hierarchical level $l = 10$). Although the Trapdoor and Test algorithms of our proposed scheme are slightly worse than [4], the running time of these algorithms are much faster than those of Key generation and Ciphertext generation algorithms in [4, 5]. Therefore, our proposed scheme is more efficient than the dHPEKS schemes of [4, 5]. The running times of the Key generation, Ciphertext generation, Trapdoor, and Test algorithms of our proposed scheme are reduced by 51.78%, 7.06%, 45.61%, and 32.32%, respectively, compared to [9]. Table 1 and Figure 3 show that the running times of the Key generation, Trapdoor, and Test algorithms of our proposed scheme have fallen by 51.78%, 49.19%, and 44.23%, respectively, compared to [21]. The running times of the Ciphertext generation of our proposed scheme is almost the same as [21] and [3]. The computation costs of our proposed scheme in the Key generation, Trapdoor, and Test phases are significantly reduced by 98.25%, 87.00%, and 44.23%, respectively, in comparison with Wang *et al.*'s scheme [3], which shows that our proposed scheme is much more efficient than [3].

In order to compare the communication cost of our proposed scheme with the schemes of [3–5, 9, 21], in Table 3 and Figure 4, we count the bit length size of the ciphertexts and the trapdoors of these schemes. Let h and $|G_i|(i = 1, 2)$ denote the bit length size of the output of an ordinary hash function and the bit length size of each element of $G_i(i = 1, 2)$. We use the bit length sizes reported in [41], i.e., $h = 160 - bit$, $|G_1|= 512 - bit$ and $|G_2|= 1024 - bit$.[2] As presented in Table 3 and Figure 4, the bit length size of the ciphertexts in our proposed scheme is smaller than those in all other schemes [3–5, 9, 21]. The results show that our proposed scheme has a much lower size of trapdoor than [3, 5], and the same size as [9] and [4]. Table 3 and Figure 4 show that the bit length size of the Trapdoor in [21] has fallen by 22.92%, compared to our proposed scheme. However, the bit length size of the ciphertexts in our proposed scheme is reduced by 46.38%, compared to [21].

[2] We also count the bit length size of the output of the hash function H_4, and the integer N in [5], i.e., $\gamma_1 = 128 - bit$ (reported in [5]), $|N|= 1024-bit$ (reported in [37]), respectively.

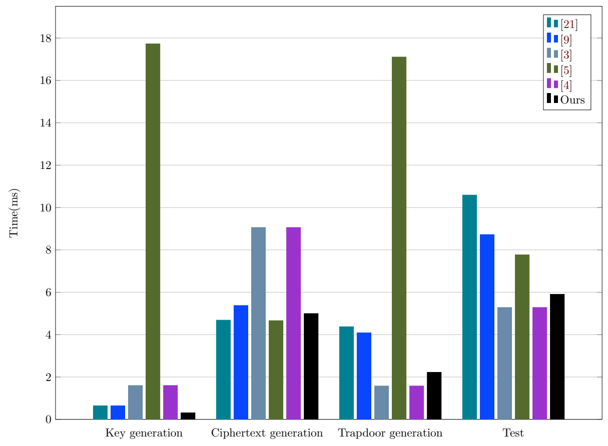

Figure 3. Computational cost comparison

Now consider a scenario where a data sender wants to encrypt a keyword for multiple data receivers in a hierarchical structure. In a PEKS scheme that does not support hierarchical access permission for users, the data sender has to encrypt the same keyword multiple times for each data receiver. However, in our proposed scheme and the schemes in [3–5], which support hierarchical access permission, the data sender only encrypts the keyword for the lowest level users of each data receiver's branch in the user's hierarchical structure. In Figure 5 and Figure 6, we compare the ciphertext generation time and the bit length size of the ciphertext of our proposed scheme with the schemes of [3–5, 9, 21] in this scenario. Figure 5 and Figure 6 show that the ciphertext generation time and the ciphertext size in [3–5] and our proposed scheme merely increase by increasing the number of different data receivers' branches in the hierarchical structure, regardless of the number of data receivers. Furthermore, in this scenario, as shown in Figure 5 and Figure 6, our proposed scheme is better than existing dHPEKS schemes [4, 5] in terms of the ciphertext generation time and the ciphertext size. In comparison with [3], the ciphertext generation time of our proposed scheme is almost the same as [3], and the ciphertext size of our proposed scheme is less than that of [3], in this scenario.

7 Conclusion

Wang *et al.* introduced the notion of hierarchical ID-based encryption with keyword search (HIBEKS) and proposed the first HIBEKS scheme in 2015 [3].

Schemes	Key generation	Ciphertext generation	Trapdoor generation	Test
[21]	$T_H + T_{sm} = 0.645$	$T_H + 6T_{sm} + T_{bp} = 4.686$	$T_H + 5T_{sm} + T_{bp} = 4.375$	$T_H + T_{sm} + 4T_{bp} = 10.589$
[9]	$T_H + T_{sm} = 0.645$	$4T_H + 5T_{sm} + T_{bp} = 5.377$	$2T_H + 3T_{sm} + T_{bp} = 4.087$	$T_H + 3T_{sm} + 3T_{bp} = 8.725$
[3]	$57T_{sm} = 17.727$	$7T_{sm} + 2T_{bp} = 4.663$	$55T_{sm} = 17.105$	$T_{sm} + 3T_{bp} = 7.769$
[5]	$3T_H + T_{sm} = 1.313$	$2T_H + 5T_{sm} + 3T_{bp} = 9.681$	$2T_H + 3T_{sm} = 1.267$	$T_{sm} + 2T_{bp} = 5.283$
[4]	$2T_H + 3T_{sm} = 1.601$	$2T_H + 3T_{sm} + 3T_{bp} = 9.059$	$T_H + 4T_{sm} = 1.578$	$T_{sm} + 2T_{bp} = 5.283$
Ours	$T_{sm} = 0.311$	$T_H + 7T_{sm} + T_{bp} = 4.997$	$2T_H + 5T_{sm} = 2.223$	$3T_{sm} + 2T_{bp} = 5.905$

Table 1. Computational cost (ms)

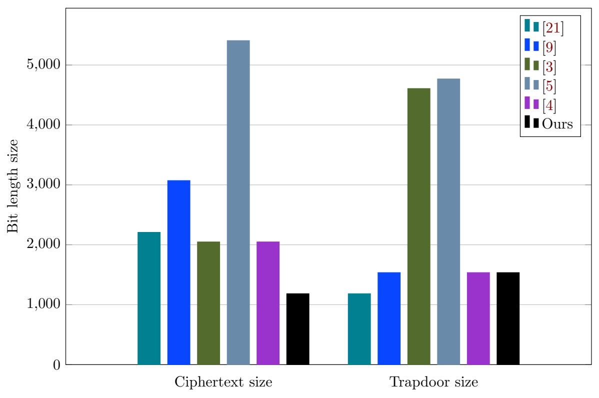

Figure 4. Communication cost comparison

	[21]	[9]	[3]	[5]	[4]	Ours
Ciphertext-indistinguishability	No	No	Yes	Yes	Yes	Yes
Security against outside KGA	Yes	Yes	No	Yes	Yes	Yes
Certificate management problem	No	No	No	Yes	Yes	No
Support hierarchical keyword search	No	No	Yes	Yes	Yes	Yes

Table 2. Security and property comparison

	Ciphertext size	Trapdoor size
[21]	$4\|G_1\|+h = 2208$	$2\|G_1\|+h = 1184$
[9]	$4\|G_1\|+\|G_2\| = 3072$	$3\|G_1\| = 1536$
[3]	$2\|G_1\|+\|G_2\| = 2048$	$9\|G_1\| = 4608$
[5]	$2\|G_1\|+4\|N\|+\gamma_1+h = 5408$	$\|G_1\|+4\|N\|+h = 4768$
[4]	$2\|G_1\|+\|G_2\| = 2048$	$3\|G_1\| = 1536$
Ours	$2\|G_1\|+h = 1184$	$3\|G_1\| = 1536$

Table 3. Communication cost comparison

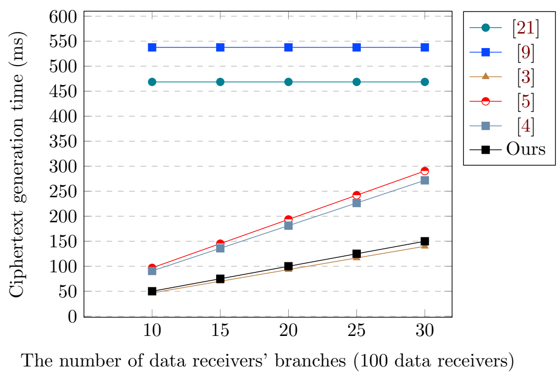

Figure 5. Ciphertext generation time along with data receivers' branches increasing

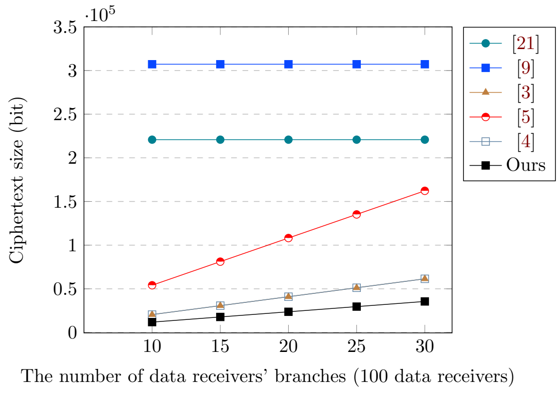

Figure 6. Ciphertext size along with data receivers' branches increasing

However, Wang *et al.*'s scheme is not secure against offline KGA. In 2020, Li *et al.* [4] introduced hierarchical structures into designated-server public key encryption with keyword search schemes and proposed the first hierarchical designated-server public key encryption with keyword search (dHPEKS) scheme. Another dHPEKS scheme was proposed in 2021 by Liu *et al.* [5]. However, the schemes of [4] and [5] suffer from the certificate management problem which is inherent to the cryptographic schemes in the PKI setting. The main purpose of this paper is to devise techniques to avoid the certificate management problem that exists in the public key setting while providing necessary security requirements in PEKS and supporting hierarchical structures. To this end, we first introduce the notion of dHIBSE and formulate its security model. We further propose a dHIBSE scheme and prove its security under the defined security model. Moreover, we illustrate the overall superiority of our proposed scheme by comparing it with other related schemes in terms of security requirements, communication and computational costs.

Acknowledgment

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pages 44–55. IEEE, 2000.

[2] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*, pages 506–522. Springer, 2004.

[3] Xiaofen Wang, Xiaosong Zhang, and Yi Mu. Hierarchical id-based searchable encryption with constant size ciphertext in cloud. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 1024–1031. IEEE, 2015.

[4] Hongbo Li, Qiong Huang, and Willy Susilo. A secure cloud data sharing protocol for enterprise supporting hierarchical keyword search. *IEEE Transactions on Dependable and Secure Computing*, 2020.

[5] Tong Liu, Yinbin Miao, Kim-Kwang Raymond Choo, Hongwei Li, Ximeng Liu, Xiangdong Meng, and Robert H Deng. Time-controlled hierarchical multi-keyword search over encrypted data in cloud-assisted iot. *IEEE Internet of Things Journal*, 2021.

[6] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. Public key encryption with keyword search revisited. In *International conference on Computational Science and Its Applications*, pages 1249–1259. Springer, 2008.

[7] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Workshop on Secure Data Management*, pages 75–83. Springer, 2006.

[8] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software*, 83(5):763–771, 2010.

[9] Yang Lu, Gang Wang, Jiguo Li, and Jian Shen. Efficient designated server identity-based encryption with conjunctive keyword search. *Annals of Telecommunications*, 72(5):359–370, Jun 2017.

[10] Yousheng Zhou, Han Guo, Feng Wang, and Wenjun Luo. Multi-key searchable encryption with designated server. *Intelligent Automation & Soft Computing*, 22(2):295–301, 2016.

[11] Zeeshan Pervez, Ammar Ahmad Awan, Asad Masood Khattak, Sungyoung Lee, and Eui-Nam Huh. Privacy-aware searching with oblivious term matching for cloud storage. *The Journal of Supercomputing*, 63:538–560, 2013.

[12] Hyun Sook Rhee, Jong Hwan Park, and Dong Hoon Lee. Generic construction of designated tester public-key encryption with keyword search. *Information Sciences*, 205(Supplement C):93 – 109, 2012.

[13] Zhen Li, Minghao Zhao, Han Jiang, and Qiuliang Xu. Multi-user searchable encryption with a designated server. *Annals of Telecommunications*, 72(9):617–629, 2017.

[14] Yang Lu and Jiguo Li. Constructing designated server public key encryption with keyword search schemes withstanding keyword guessing attacks. *International Journal of Communication Systems*, 32(3):e3862, 2019.

[15] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Annual international cryptology conference*, pages 205–222. Springer, 2005.

[16] Xiuxia Tian and Yong Wang. Id-based encryption with keyword search scheme from bilinear pairings. In *2008 4th International Conference on Wireless Communications, Networking and Mobile Computing*, pages 1–4. IEEE, 2008.

[17] Koji Tomida, Masami Mohri, and Yoshiaki Shiraishi. Keyword searchable encryption with access control from a certain identity-based encryption. In *Future Information Technology*, pages 113–118. Springer, 2014.

[18] Li Xu, Chi-Yao Weng, Lun-Pin Yuan, Mu-En Wu, Raylin Tso, and Hung-Min Sun. A shareable keyword search over encrypted data in cloud computing. *The Journal of Supercomputing*, 74:1001–1023, 2018.

[19] Koji Tomida, Hiroshi Doi, Masami Mohri, and Yoshiaki Shiraishi. Ciphertext divided anonymous hibe and its transformation to identity-based encryption with keyword search. *Journal of information processing*, 23(5):562–569, 2015.

[20] Tsu-Yang Wu, Tung-Tso Tsai, and Yuh-Min Tseng. Efficient searchable id-based encryption

with a designated server. *annals of telecommunications - annales des télécommunications*, 69(7):391–402, 2014.

[21] Xiao-Fen Wang, Yi Mu, Rongmao Chen, and Xiao-Song Zhang. Secure channel free id-based searchable encryption for peer-to-peer group. *Journal of Computer Science and Technology*, 31(5):1012–1027, 2016.

[22] Mahnaz Noroozi and Ziba Eslami. Public key authenticated encryption with keyword search: revisited. *IET Information Security*, 13(4):336–342, 2018.

[23] Dong Zhang, Qing Fan, Hongyi Qiao, and Min Luo. A public-key encryption with multi-keyword search scheme for cloud-based smart grids. In *2021 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 1–6. IEEE, 2021.

[24] Tolun Tosun and Erkay Savaş. Fsds: A practical and fully secure document similarity search over encrypted data with lightweight client. *Journal of Information Security and Applications*, 59:102830, 2021.

[25] Mimi Ma, Debiao He, Shuqin Fan, and Dengguo Feng. Certificateless searchable public key encryption scheme secure against keyword guessing attacks for smart healthcare. *Journal of Information Security and Applications*, 50:102429, 2020.

[26] Nasrollah Pakniat, Danial Shiraly, and Ziba Eslami. Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial iot. *Journal of Information Security and Applications*, 53:102525, 2020.

[27] Sanjeet Kumar Nayak and Somanath Tripathy. Seps: Efficient public-key based secure search over outsourced data. *Journal of Information Security and Applications*, 61:102932, 2021.

[28] Yu Zhang, Yin Li, and Yifan Wang. Secure and efficient searchable public key encryption for resource constrained environment based on pairings under prime order group. *Security and Communication Networks*, 2019, 2019.

[29] Ahmad Akmal Aminuddin Mohd Kamal and Keiichi Iwamura. Searchable encryption using secret sharing scheme that realizes direct search of encrypted documents and disjunctive search of multiple keywords. *Journal of Information Security and Applications*, 59:102824, 2021.

[30] Ming-Fong Tsai and Yi-Hong Wu. User intent prediction search engine system based on query analysis and image recognition technologies. *The Journal of Supercomputing*, pages 1–33, 2022.

[31] Danial Shiraly, Nasrollah Pakniat, Mahnaz Noroozi, and Ziba Eslami. Pairing-free certificateless authenticated encryption with keyword search. *Journal of Systems Architecture*, page 102390, 2022.

[32] Yang Lu and Jiguo Li. Efficient searchable public key encryption against keyword guessing attacks for cloud-based emr systems. *Cluster Computing*, 22(1):285–299, 2019.

[33] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. On the integration of public key data encryption and public key encryption with keyword search. In *International Conference on Information Security*, pages 217–232. Springer, 2006.

[34] Yong Ho Hwang and Pil Joong Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In *International conference on pairing-based cryptography*, pages 2–22. Springer, 2007.

[35] Mimi Ma, Shuqin Fan, and Dengguo Feng. Multi-user certificateless public key encryption with conjunctive keyword search for cloud-based telemedicine. *Journal of Information Security and Applications*, 55:102652, 2020.

[36] Xueqiao Liu, Kai He, Guomin Yang, Willy Susilo, Joseph Tonien, and Qiong Huang. Broadcast authenticated encryption with keyword search. In *Australasian Conference on Information Security and Privacy*, pages 193–213. Springer, 2021.

[37] Ximeng Liu, Robert H Deng, Kim-Kwang Raymond Choo, and Jian Weng. An efficient privacy-preserving outsourced calculation toolkit with multiple keys. *IEEE Transactions on Information Forensics and Security*, 11(11):2401–2414, 2016.

[38] Ziba Eslami and Nasrollah Pakniat. Certificateless aggregate signcryption: Security model and a concrete construction secure in the random oracle model. *Journal of King Saud University - Computer and Information Sciences*, 26(3):276 – 286, 2014.

[39] Benoît Libert and Jean-Jacques Quisquater. Identity based undeniable signatures. In *Cryptographers' track at the RSA conference*, pages 112–125. Springer, 2004.

[40] Yong Yu, Chunxiang Xu, Xiaosong Zhang, and Yongjian Liao. Designated verifier proxy signature scheme without random oracles. *Computers & Mathematics with Applications*, 57(8):1352–1364, 2009.

[41] Yang Lu and Jiguo Li. Efficient searchable public key encryption against keyword guessing attacks for cloud-based emr systems. *Cluster Computing*, 22(1):285–299, 2019.

[42] B Lynn. Pbc library: the pairing-based cryptography library. *http://crypto.stanford.edu/pbc/ (2013). Accessed 1 April 2014*, 2013.

Danial Shiraly received the B.Sc. degree in Mathematics from Farhangian University and M.Sc. degree in Computer Science from Shahid Beheshti University in 2018 and 2020, respectively. His research interests include Cryptography and Information Security.

Nasrollah Pakniat has a Ph.D. in Mathematics, graduated in 2015 from Shahid Beheshti University. He received his M.Sc. degree in Computer Science from Shahid Beheshti University in 2011. He holds a B.Sc. degree in Computer Science from Shahid Bahonar University of Kerman in 2008. He began his scientific experience in 2016 as a faculty member in IranDoc. Now he is an assistant professor of the Information Science Research Center. His efforts at Irandoc has mainly devoted to development of Cryptographic Protocols and Text Mining algorithms. His research interests include Cryptography, Network Security and Text Mining.

Ziba Eslami received her Ph.D. in Applied Mathematics from Tehran University in 2000. During the academic years 2000–2003, she was a postdoctoral fellow in the Institute for Research in Fundamental Sciences (IPM). She served as a non-resident researcher at IPM during 2003–2005. Currently, she is an associate professor in the Department of Data and Computer Science at Shahid Beheshti University (SBU) in Iran. Her research is primarily centered around Cryptography and Security of Cryptographic Protocols.