Hindawi
Security and Communication Networks
Volume 2019, Article ID 4268731, 16 pages
https://doi.org/10.1155/2019/4268731

Research Article

A Secure Data Sharing Scheme with Designated Server

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

[1]*School of Mathematics, Shandong University, Jinan, Shandong 250100, China*
[2]*State Key Laboratory of Information Security Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[3]*Centre for Computer and Computational Science at School of Computing and Mathematical Sciences, University of Greenwich, London, UK*

Correspondence should be addressed to Jing Qin; qinjing@sdu.edu.cn

Received 29 March 2019; Revised 22 May 2019; Accepted 4 June 2019; Published 27 June 2019

Academic Editor: Kuo-Hui Yeh

The cloud-assisted Internet of Things (CIoT) is booming, which utilizes powerful data processing capabilities of the cloud platform to solve massive Internet of Things (IoT) data. However, the CIoT faces new security challenges, such as the confidentiality of the outsourced data. Data encryption is a fundamental technique that can guarantee the confidentiality of outsourced data, but it limits target encrypted data retrieval from cloud platform. Public key encryption with keyword search (PEKS) provides a promising solution to address this problem. In PEKS, a cloud server can be authorized to search the keyword in encrypted documents and retrieve associated encrypted documents for the receiver. However, most existing PEKS schemes merely focus on keyword search function while ignoring the associated documents encryption/decryption function. Thus, in practice, a PEKS scheme must cooperate with another separated public key encryption (PKE) scheme to fulfill a completely secure data sharing scheme. To address this problem, in this paper, we propose a secure data sharing scheme with designated server that combines PKE scheme with PEKS scheme, which provides both keyword search and documents encryption/decryption functions. Furthermore, only the designated server can search the keyword via encrypted documents for enhanced security in our work. Moreover, our scheme also satisfies the public verifiability of search results, which includes both keywords and documents ciphertexts' correctness and integrity. As to the security, our scheme provides stronger indistinguishability security of document and keyword in the proposed security model.

1. Introduction

Cloud storage has been widely deployed in daily life. As a promising application, cloud-assisted Internet of Things (CIoT) have utilized cloud storage to store their data to reduce the burden of data processing, as shown in Figure 1. In CIoT, users rely on the cloud platform to complete the data storage and data sharing. Generally, data is migrated from the user to a cloud server, in which the cloud server is widely recognized as an honest-but-curious party. However, the cloud storage is provided by a third party and the user's data may have private information. Therefore, the user should encrypt the data prior to uploading it to cloud server for protecting data confidentiality. Unfortunately, this approach eliminates the data search service provided by modern search engines, which inevitably makes the effective data search function become a challenging research problem. A trivial solution is that we download the full encrypted data and then decrypt it to obtain the plaintext data. Of course, it needs to occupy a large amount of local storage space and communication consumption. Another trivial solution is that the user sends the private key to cloud server. The server decrypts the encrypted data in the cloud, searches for the user in plaintext, and retrieves the intended data. This solution solves the above problem, but it compromises data privacy, which violates the original intention of data encryption. Focusing on the aforementioned problem, searchable encryption was proposed. Searchable encryption enables a data receiver to authorize the cloud server to search in encrypted documents, where encrypted documents are not needed to be decrypted. Searchable encryption is mainly divided into two techniques, which are symmetric searchable encryption (SSE) and public key encryption with keyword search (PEKS). In SSE, a shared key is required to achieve data sharing function in the cloud

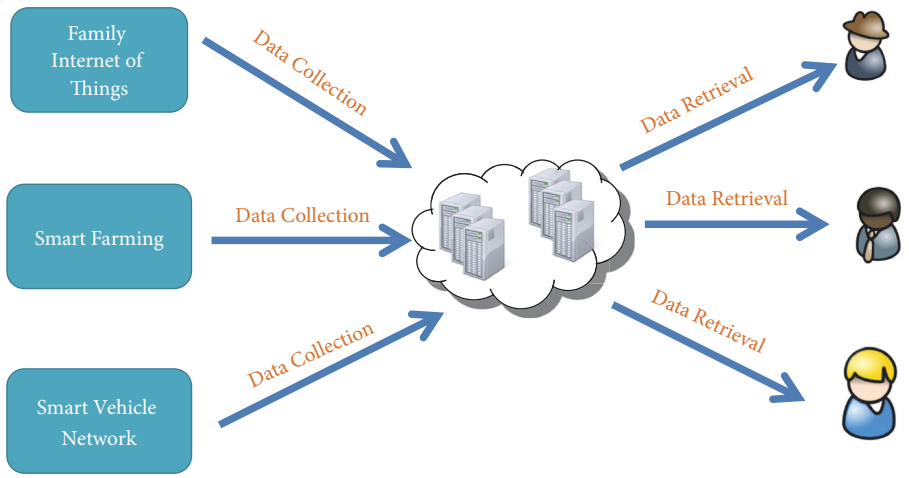

Figure 1: Functions of a cloud-assisted IoT.

platform. PEKS was proposed in 2004 by Boneh et al. [1], which can realize the keyword search function and eliminate complicated key management in cloud platform.

The general PEKS system includes three participants, i.e., a data sender, a data receiver, and a cloud server. Data sender encrypts keywords index using receiver's public key and uploads keyword ciphertexts to cloud server. The data receiver uses its private key to generate a keyword trapdoor and transmits trapdoor to cloud server. Cloud server uses the trapdoor to match the keyword ciphertext; if the keyword in the ciphertext and the keyword in the trapdoor are identical, its outputs are equal; otherwise, its outputs are not equal.

However, PEKS mainly focuses on the keyword search process and omits the associated documents encryption/decryption process, which only explains using a standard encryption scheme to encrypt associated documents. However, in the actual applications, the documents encryption/decryption is indispensable, since the corresponding documents are what we really need. Therefore, it is essential and meaningful to combine the public key encryption (PKE) scheme with the PEKS scheme to form a completely secure data sharing scheme. For this reason, a completed scheme named integrated PKE and PEKS scheme is suggested, which combines PEKS with PKE to encrypt keyword w and its corresponding document message m together. The PEKS-PKE system includes three participants, i.e., a data sender, a data receiver, and a cloud server. Data sender encrypts message m using receiver's public key pk_r. It also encrypts keyword w_i with receiver's public key pk_r and appends to the resulting message ciphertext and then uploads encrypted message and keywords $C_m \parallel C_{w_i}$ to cloud server. Receiver generates the trapdoor T_w with its private key sk_r and uploads T_w to cloud server. The cloud server matches the keyword trapdoor T_w with encrypted keywords C_{w_i}, and if the same keyword is used, it outputs *yes* and returns C_m to receiver; else, it outputs *no* and returns $\perp$. The data receiver uses its private key sk_r to decrypt the message ciphertext C_m.

This integrated scheme PEKS-PKE can provide keyword w search function and message m encryption/decryption function. Most of PEKS schemes after Boneh's scheme do not introduce how to achieve this completed scheme.

When searching the keyword in the cloud, we generally assume that the cloud server is honest but curious, which means that it performs the search operation honestly; actually it is curious about the keyword content. However, in practical applications, the server may not always behave honestly. Generally, the cloud server is managed and operated by a business company. The company may delete encrypted data for their benefits for releasing storage space. In addition, the server may be broken into by malicious intruder Eve or it may unintentionally delete data. When performing a search operation, the server may return the part of the search results to deceive the receiver. Since the receiver does not know the content of the encrypted document, or even whether the encrypted document is associated with the keyword; this poses a threat to the receiver's data correctness and integrity. In addition to, the receiver may declare that the cloud server has lost some data or returned incorrect search results and doubt about the behavior of cloud server even if the provider has performed all required operations honestly. This will cause disputes. Therefore, if there is an honest-but-curious third party who can verify the integrity and correctness of the search results in a public manner, we can entrust the cloud server honestly performs the keyword search operation and solves the dispute with receiver.

Considering a specific scenario, Personal Health Record (PHR) is confidential documens to anyone except the patient and the chief physician. In order to protect patients' privacy, patients need to encrypt the PHR data prior to uploading it to cloud server. We can use a PEKS scheme to solve keyword search problem in encrypted PHR. However, the above PEKS requires that the cloud server is totally trusted; that is, it honestly stores the encrypted documents, performs the search operation, and returns the encrypted PHR. We know that it is not practical to assume a cloud platform is honest for a hospital. Generally, the hospital will outsource the construction of the cloud platform to a professional company and the physical control of the encrypted PHR belongs to

professional company. Therefore, the cloud platform may delete some encrypted PHR to release their storage space for their economic benefits. They may want to perform the dishonest search operation to deceive the chief physician and the patient. The chief physician does not know the correctness and integrity of the search results, and it is in charge of the diagnosis and treatment of patients. When the encrypted PHR search results are incomplete, the chief physician may make incorrect diagnostic results with serious consequences. Furthermore, once there is a medical negligence, the chief physician may unilaterally declare that the cloud server has lost some data or returned incorrect search results and deliberately pass the buck to the cloud service provider even if the provider has performed all required operations honestly. A straightforward solution is that the patient sends the PHR integrity evidence to the chief physician. The chief physician downloads the total encrypted data and then checks the integrity. This solution has two drawbacks. One drawback is that it breaks the asymmetry of PEKS, and on the other hand it is too expensive for the communication cost.

From the above discussion, we can know that it is importance of combining PKE and PEKS with providing the public verifiability under the untrusted cloud platform. Maintaining the privacy and public verifiability of search results are exciting and unresolved research problems.

Recently, Zhang et al. [2] proposed a public verifiable searchable encryption scheme. The scheme discussed the correctness of the returned keyword ciphertexts and lacked the integrity of both documents and keywords ciphertexts.

In 2018, we proposed a PEKS with public verifiability scheme [3]. Our scheme achieves the correctness and integrity of keywords ciphertext. However, when the cloud server is not honest, we only discuss keywords ciphertext is incomplete, since the corresponding encrypted documents are what we really need. That is to say, when cloud server returns encrypted document, it also may return another uncorrelated encrypted document with a corrected keyword ciphertext. In this case, the cloud server can also pass the verification by sweeping attack. In addition, the taggen phase lacks an index label and it will reduce the efficiency of verification. Furthermore, the tag only includes the keyword and the corresponding document serial number, in which it may also let cloud server easily forge the tag. Therefore, in practical applications, the scheme is very fragile. Our previous scheme also only focuses on keyword search function, while ignoring the associated documents encryption/decryption function.

In this paper, we propose a secure data sharing scheme with designated server that captures both functions of PKE and PEKS, which provides the functions both keyword search and documents encryption/decryption. Furthermore, only the designated server can search the keyword for enhanced security. Our scheme can also satisfy the public verifiability of search results, including ciphertexts both documents and keywords, which achieves the correctness and integrity. The scheme is great improvement comparing our previous work.

1.1. Our Contributions. Specifically, our contributions are as follows:

(1) We introduce the definition of secure data sharing scheme with designated server, which satisfies the functions both of keyword search and documents encryption/decryption.

(2) We propose a special secure data sharing scheme with designated server. Our scheme achieves security of the document indistinguishability against chosen ciphertext attack (IND-CCA), keyword indistinguishability against chosen keyword attack (IND-CKA), and trapdoor indistinguishability.

(3) It achieves the public verifiability of search results, which includes both keywords and documents ciphertexts' correctness and integrity.

(4) Our scheme removes the secure channel between the data receiver and the cloud server, which only the designated server can perform matching operation.

Technical route is as follows: we choose the Variable Hashed Elgamal scheme [4] as document encryption/decryption scheme and the PEKS with a designated tester (dPEKS) scheme as the searchable encryption scheme. These two schemes are basic components. Variable Elgamal scheme is an encryption scheme that provides document ciphertext indistinguishability security against chosen ciphertext attack (IND-CCA). The dPEKS scheme is an searchable encryption scheme that provides keyword ciphertext indistinguishability security against chosen keyword attack(IND-CKA). The trivial solution is to combine Variable Elgamal encryption with dPEKS, but there is a problem that the server is not fully trusted. The server may perform swapping keyword attacks. The swapping keyword attack is that the data receiver gets the document message it does not need. For example, when the receiver searches the document m_1 corresponding to the keyword w_1, the server sweeps the documents m_2, m_1 and returns the document m_2 corresponding to the keyword w_2. Therefore, the receiver can not get the document m_1 that it really needs. To resist this attack, we need to bind the keyword ciphertext and document ciphertext so that the malicious server cannot perform the sweeping attacks. In addition to, we also need to consider the security of combined scheme.

Since the widely used keyword space is limited, the outside adversary can guess a keyword and generate the keyword ciphertext; if the outside adversary can perform the matching operation, it can get the keyword in the trapdoor until guess the true keyword. We call this attack named offline keyword guessing attack (offline KGA). Therefore, to resist this offline KGA, we generate a key pair for the cloud server. Only the designated server can perform the keyword search operation to avoid outside adversary's offline KGA. What we need to point out here is that the adversary may get the guessing keyword by comparing two bilinear pairs without generating keyword ciphertext. We also need the trapdoor satisfies trapdoor indistinguishability in proposed scheme to resist offline KGA.

1.2. Related Works. Song et al. [8] proposed the first symmetric searchable encryption (SSE) in 2000. Song's scheme requires a word-by-word comparison to complete the keyword search operation. After Song's work, many researchers propose SSE schemes [9–11].

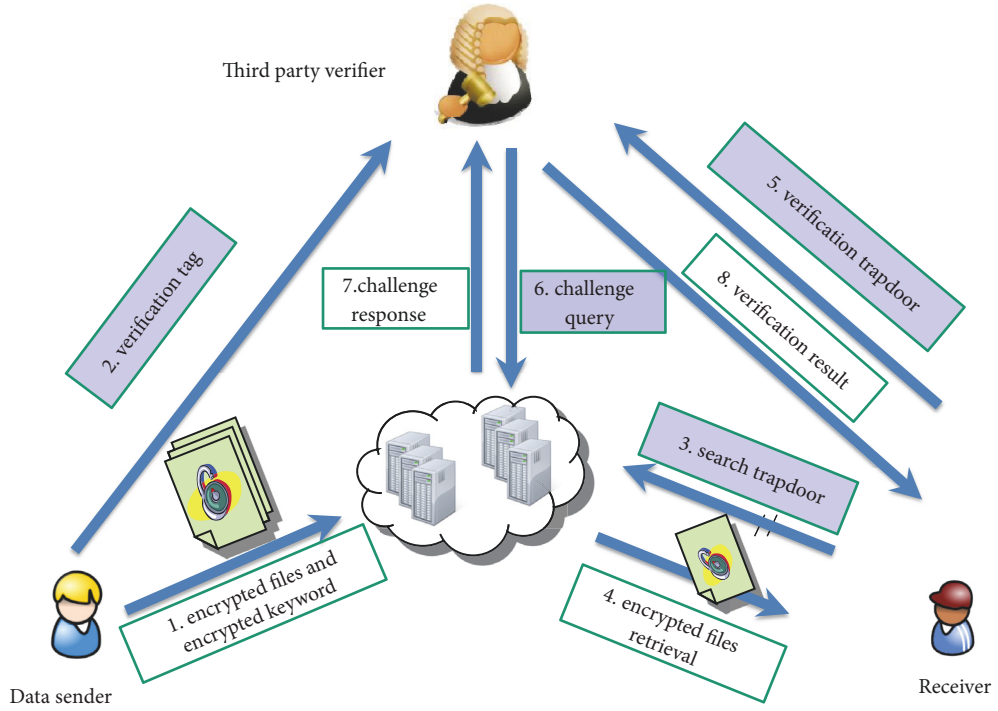

Figure 2: Secure data sharing scheme with designated server model.

Boneh et al. [1] proposed the first PEKS scheme in 2004. Boneh et al.'s scheme realizes the data sharing and the keyword search functions. In 2005, Abdalla et al. explored the conversion relationship between identity based encryption (IBE) and PEKS [12]. It is shown that an anonymous IBE scheme could be transformed into a PEKS scheme and it proposed a temporary keyword search scheme. In 2008, Baek et al. [5] proposed the PEKS with a designated tester scheme that does not require a secure channel. In 2010, Rhee et al. [13] proposed a scheme that can resist outside attacker's offline KGA in trapdoor indistinguishability security model. In 2013, Fang et al. [14] proposed a PEKS scheme in the standard model, which can resist outside attacker's offline KGA. Later, many researchers studied the offline KGA and proposed many schemes [15–20].

In 2014, Zheng et al. [21] proposed the first verifiable attribute-based encryption with keyword search scheme. This scheme flexibly uses bloom filters and signatures to achieve the verifiability. In Zheng et al. scheme, attribute ciphertext and trapdoor are proportional to the number of attributes. It also requires a secure channel and supports the receiver private verification. After this work, many researchers proposed attribute-based encryption with keyword search schemes [22, 23].

In 2006, Baek et al. [24] proposed the first PEKS-PKE scheme. This scheme realizes the functions of document encryption/decryption and keyword search. Baek et al.'s scheme only discusses the security of document and does not involve the security of keyword, and it needs a secure channel. In 2009, Zhang et al. [7] proposed a PEKS-PKE scheme, which involves the security of keyword. It has two public and private key pairs and requires a secure channel. Chen et al. [6] proposed a PEKS-PKE scheme in 2016 and Chen et al.'s general construction leaks the keyword to the server. These three schemes do not discuss the security of trapdoor, and their schemes do not support the correctness and integrity verification of the search results, including the returned ciphertexts both documents and keywords.

1.3. Organization. The paper is organized as follows. Section 1 is the introduction. The scheme definition and security models are described in Section 2. A secure data sharing scheme with designated server is proposed in Section 3. We analyze the security and efficiency of the proposed scheme in Section 3. The paper is concluded in Section 4.

2. Scheme Definition and Security Models

2.1. System Model. The system model of secure data sharing scheme with designated server that supporting public verifiability is shown in Figure 2: there are four participants in this model including a data sender, a receiver, cloud server, and the third party verifier.

First of all, data sender encrypts the document message m by using receiver's public key pk_r and encryption algorithm enc to form message ciphertext C_j, encrypts the corresponding keyword w_i by using cloud server's public key pk_s, receiver's public key pk_r, and encryption algorithm $peks$ to form keyword ciphertext C_{w_i}, then binds the C_j and C_{w_i} to form ciphertext u, and uploads the ciphertext u to cloud server; data sender also uses the cloud server public key pk_s, the receiver public key pk_r, the keyword w_i, the keyword

ciphertext C_{w_i}, and corresponding message ciphertext C_j to generate the verification tag τ and sends the verification tag τ to third party verifier.

Secondly, the receiver uses its secret key sk_r and cloud server's public key pk_s to generate keyword trapdoor T_w with the verification trapdoor VK_w and transmits T_w to cloud server. And then, cloud server uses the trapdoor T_w and ciphertext u to compute the matching result. If the keyword in the ciphertext u and the keyword in the trapdoor T_w are equal, cloud server returns the ciphertext u to receiver. Next, to obtain the message m, the receiver decrypts the u by using its secret key sk_r.

In final step, when there is a dispute between cloud server and receiver, the third party verifier uses the verification trapdoor VK_w from receiver, the verification tag τ, and returns the verification result.

2.2. Scheme Definition

Definition 1. More specifically, a secure data sharing scheme with designated server consists of the following algorithms:

(1) $sp \longleftarrow \mathsf{SysGen}(1^k)$: on input a security parameter 1^k and output a system parameter sp.

(2) $(\mathsf{pk_r}, \mathsf{sk_r}) \longleftarrow \mathsf{KeyGen_{receiver}}(\mathsf{sp})$: on input the system parameter sp and output a pair of public and secret key (pk_r, sk_r) for the receiver.

(3) $(\mathsf{pk_s}, \mathsf{sk_s}) \longleftarrow \mathsf{KeyGen_{server}}(\mathsf{sp})$: on input the system parameter sp and output a pair of public and secret key (pk_s, sk_s) for the cloud server.

(4) $\mathsf{u} \longleftarrow \mathsf{Enc}(\mathsf{sp}, \mathsf{pk_s}, \mathsf{pk_r}, \mathsf{w}, \mathsf{m})$: on input the system parameter sp, the cloud server public key pk_s, the receiver public key pk_r, the keyword w, the document message m and output a ciphertext $u = (C_j, C_w, \sigma)$, which $C_j = enc(sp, pk_r, m)$ is message ciphertext, $C_w = peks(sp, pk_s, pk_r, w)$ is keyword ciphertext, σ is a binding tag.

(5) $\tau_{\mathsf{w_i}} \longleftarrow \mathsf{Tag}(\mathsf{sp}, \mathsf{pk_s}, \mathsf{pk_r}, \mathsf{w_i}, \mathsf{C_{w_i}}, \mathsf{C_j})$: on input the system parameter sp, the cloud server public key pk_s, the receiver public key pk_r, the keyword w_i, the keyword ciphertext C_{w_i} and the message ciphertext C_j, which i is the serial number of the keyword and j is the message serial number corresponded the keyword w_i, and output the verification tag τ_{w_i}, releases τ_{w_i} to the third party verifier.

(6) $(\mathsf{T_w}, \mathsf{VK_w}) \longleftarrow \mathsf{Trapdoor}(\mathsf{sp}, \mathsf{pk_s}, \mathsf{sk_r}, \mathsf{w})$: on input the system parameter sp, the cloud server public key pk_s, the receiver secret key sk_r, the keyword w, and output the keyword search trapdoor T_w, the verification trapdoor VK_w.

(7) $\mathsf{u\ or\ } \perp \longleftarrow \mathsf{Test}(\mathsf{sp}, \mathsf{sk_s}, \mathsf{T_w}, \mathsf{u})$: on input the system parameter sp, the cloud server secret key sk_s, the keyword search trapdoor T_w, the ciphertext u, and output ciphertext u if the keyword in the u and the keyword in the T_w are equal; otherwise, output $\perp$.

(8) $\mathsf{m\ or\ } \perp \longleftarrow \mathsf{Dec}(\mathsf{sp}, \mathsf{sk_r}, \mathsf{u})$: on input the system parameter sp, the receiver secret key sk_r, the ciphertext u and output the message m or $\perp$.

(9) $\mathsf{1\ or\ 0} \longleftarrow \mathsf{Public\ verify}(\mathsf{sp}, \mathsf{VK_w}, \tau)$: when there is a dispute between cloud server and receiver, the third party verifier inputs the system parameter sp, the verification trapdoor VK_w, the verification tag τ, output the verification result 1 if satisfies condition; and 0 otherwise.

$Kset \longleftarrow \phi$
$(pk_s, sk_s, pk_r, sk_r) \longleftarrow KeyGen(sp)$;
$(w_0, w_1, m^*) \longleftarrow \mathscr{A}_1^{\mathscr{O}_{1,2}}(sp, (pk_s, sk_s), pk_r)$;
$u_b \longleftarrow \mathscr{B}(m^*, w_b, pk_s, pk_r, b \in \{0,1\})$;
$b' \longleftarrow \mathscr{A}_1^{\mathscr{O}_{1,2}}(u_b, guess)$;
if $\{w_0, w_1\} \bigcap Kset = \phi$, then return 1, if $b' = b$;
else return 0.
Oracle $\mathscr{O}_1(w)$:
$Kset = Kset \bigcup \{w\}, T_w \longleftarrow \mathscr{O}(pk_s, sk_r, w)$;
return $\{T_w\}$
Oracle $\mathscr{O}_2(u)$:
$m \longleftarrow \mathscr{O}_2(sk_r, u)$;
return $\{m\}$

Box 1: Game IND-CKA 1 $Exp_{\mathscr{A}_1}^{CKA}$.

2.3. Security Model.

We define four security models, including the security models of IND-CKA, trapdoor indistinguishability (IND-Trapdoor), IND-CCA, and the public verifiability.

We define the keyword ciphertext semantic security. Any adversary can not distinguish the challenge keyword ciphertext unless the trapdoor is available. Formally, we define security game IND-CKA i played between a challenger $\mathscr{B}$ and adversary $\mathscr{A}_i, i = 1, 2$.

In IND-CKA 1, the challenger $\mathscr{B}$ generates the receiver key pair (pk_r, sk_r) and sends public key pk_r to the cloud server adversary $\mathscr{A}_1$. The adversary $\mathscr{A}_1$ generates the cloud server key pair (pk_s, sk_s) and sends public key pk_s to the challenger. The adversary can access the trapdoor oracle $\mathscr{O}_1(w)$ to get any keyword trapdoor w_i and access the decryption oracle $\mathscr{O}_2(u)$ on any ciphertexts u and then outputs two distinct challenge keywords and a message (w_0, w_1, m^*), in which $w_b \neq w_i, b \in \{0,1\}$. The challenger generates challenge ciphertext u_b of (w_b, m^*) with a random bit b and sends it to $\mathscr{A}_1$. During the game, the adversary can adaptively continue the query to decryption oracle $\mathscr{O}_2(u)$ and trapdoor oracle $\mathscr{O}_1(w)$ unless the challenge keywords. Finally, the adversary $\mathscr{A}_1$ outputs a bit b' as its guess.

In IND-CKA 2, the game played between a challenger $\mathscr{B}$ and an outside adversary $\mathscr{A}_2$ is similar to IND-CKA 1. The details are in the following definition.

Definition 2 ((IND-CKA) see Boxes 1 and 2). A secure data sharing scheme with designated server is IND-CKA secure if no probabilistic polynomial time (PPT) adversary $\mathscr{A}_1$ can win game IND-CKA 1 and PPT adversary $\mathscr{A}_2$ can win game IND-CKA 2 with nonnegligible advantage, where $\mathscr{B}$ is the challenger, $\mathscr{A}_1$ is cloud server, and $\mathscr{A}_2$ is the outside adversary (including a receiver).

We define A_i advantage as

$$Adv_{\mathscr{A}_i}^{IND-CKA} = \left| Pr\left[b = b'\right] - \frac{1}{2} \right|, \quad i \in \{1, 2\}. \tag{1}$$

Next, we define the keyword trapdoor semantic security. Any adversary can not distinguish the challenge trapdoor; that is to say, the challenge trapdoor does not reveal any

$(pk_s, sk_s, pk_r, sk_r) \longleftarrow KeyGen(sp)$;
$(w_0, w_1, m^*) \longleftarrow \mathcal{A}_2^{\mathcal{O}_2}(sp, (pk_r, sk_r), pk_s)$;
$u_b \longleftarrow \mathcal{B}(w_b, m^*, pk_s, pk_r, b \in \{0,1\})$;
$b' \longleftarrow \mathcal{A}_2(u_b, guess)$;
then return 1, if $b' = b$;
else return 0.
Oracle $\mathcal{O}_2(u)$:
$m \longleftarrow \mathcal{O}_2(sk_r, u)$;
return $\{m\}$

Box 2: Game IND-CKA 2 $Exp_{\mathcal{A}_2}^{CKA}$.

$Kset \longleftarrow \phi$
$(pk_s, sk_s, pk_r, sk_r) \longleftarrow KeyGen(sp)$;
$(w_0, w_1) \longleftarrow \mathcal{A}_3^{\mathcal{O}_1}(sp, pk_s, pk_r)$;
$T_b \longleftarrow \mathcal{B}(w_b, pk_s, sk_r, b \in \{0,1\})$;
$b' \longleftarrow \mathcal{A}_3^{\mathcal{O}_1}(T_b, guess)$;
if $\{w_0, w_1\} \bigcap Kset = \phi$, then return 1, if $b' = b$;
else return 0.
Oracle $\mathcal{O}_1(w)$:
$Kset = Kset \bigcup \{w\}$, $T_w \longleftarrow \mathcal{O}_1(pk_s, sk_r, w)$;
return $\{T_w\}$

Box 3: Game IND-Trapdoor $Exp_{\mathcal{A}_3}$.

information about the keyword. The IND-Trapdoor is similar to the IND-CKA 1. The adversary is given the challenge trapdoor instead of the challenge ciphertext. In IND-Trapdoor security model, the challenger $\mathcal{B}$ generates two key pairs $(pk_s, sk_s), (pk_r, sk_r)$ and sends public keys pk_s, pk_r to the adversary $\mathcal{A}_3$. $\mathcal{A}_3$ can access the trapdoor oracle $\mathcal{O}_1(w)$ to get any keyword trapdoor w_i and outputs two distinct challenge keywords (w_0, w_1), in which $w_b \neq w_i, b \in \{0,1\}$. The challenger generates challenge trapdoor T_b of w_b with a random bit b and sends it to $\mathcal{A}_3$. During the game, the adversary $\mathcal{A}_3$ can adaptively continue to query trapdoor oracle $\mathcal{O}_1(w)$ unless the challenge keywords. Finally, the adversary $\mathcal{A}_3$ outputs b' as its guess.

Definition 3 ((IND-Trapdoor); see Box 3). A secure data sharing scheme with designated server satisfies trapdoor indistinguishability if no PPT adversary $\mathcal{A}_3$ can win the game IND-Trapdoor with nonnegligible advantage, where $\mathcal{B}$ is the challenger and $\mathcal{A}_3$ is an outside adversary.

We define $\mathcal{A}_3$ advantage as

$$Adv_{\mathcal{A}_3}^{IND-Trapdoor} = \left| Pr\left[b = b'\right] - \frac{1}{2} \right|. \quad (2)$$

After that, we define the document message ciphertext semantic security. Any adversary can not distinguish the challenge message ciphertext; even it can access the decryption oracle $\mathcal{O}_2(u)$. Formally, we define security game IND-CCA.

The IND-CCA is similar to the IND-CKA 1. The difference is that the adversary outputs two distinct challenge

$(pk_s, sk_s, pk_r, sk_r) \longleftarrow KeyGen(sp)$;
$(m_0, m_1, w^*) \longleftarrow \mathcal{A}_4^{\mathcal{O}_{1,2}}(sp, (pk_s, sk_s), pk_r)$;
$u_b \longleftarrow \mathcal{B}(w^*, m_b, pk_s, pk_r, b \in \{0,1\}, w^* \neq m_0, w^* \neq m_1)$;
$b' \longleftarrow \mathcal{A}_4^{\mathcal{O}_{1,2}}(u_b, guess)$;
Then return 1, if $b' = b$;
else return 0.
Oracle $\mathcal{O}_1(w)$:
$T_w \longleftarrow \mathcal{O}(pk_s, sk_r, w)$;
return $\{T_w\}$
Oracle $\mathcal{O}_2(u)$:
$m \longleftarrow \mathcal{O}_2(sk_r, u)$;
If $u \neq u_b$, return$\{m\}$, else return $\perp$.

Box 4: Game IND-CCA $Exp_{\mathcal{A}_4}^{CCA}$.

message and a keyword (m_0, m_1, w^*). The challenger generates challenge ciphertext u_b of (w^*, m_b) with a random bit $b \in \{0,1\}$ and sends it to $\mathcal{A}_4$. During the game, the adversary can adaptively continue the query to trapdoor oracle $\mathcal{O}_1(w)$ and decryption oracle $\mathcal{O}_2(u)$ unless the challenge ciphertext u_b. We omit the details here.

Definition 4 ((IND-CCA); see Box 4). A secure data sharing scheme with designated server is IND-CCA secure if no PPT adversary $\mathcal{A}_4$ can win the game IND-CCA with nonnegligible advantage, where $\mathcal{B}$ is the challenger and $\mathcal{A}_4$ is an adversary (including the server).

We define $\mathcal{A}_4$ advantage as

$$Adv_{\mathcal{A}_4}^{IND-CCA} = \left| Pr\left[b = b'\right] - \frac{1}{2} \right|. \quad (3)$$

Finally, we define the public verifiability security. Any adversary can not forge a challenge response without the complete ciphertext.

In public verifiability security model, the challenger generates key pairs $(pk_s, sk_s), (pk_r, sk_r)$ and sends public keys pk_s, pk_r and secret key sk_s to the adversary $\mathcal{A}_5$. The challenger $\mathcal{B}$ generates a query request ch_w and sends it to adversary. The adversary $\mathcal{A}_5$ forges challenge response R' and sends it to challenger. Finally, the challenger outputs a bit b as its verification result.

Definition 5 (see Box 5). A secure data sharing scheme with designated server satisfies public verifiability security if no PPT adversary $\mathcal{A}_5$ can win the game public verifiability (PV) with nonnegligible advantage, where $\mathcal{B}$ is the challenger and $\mathcal{A}_5$ is the cloud server.

We define $\mathcal{A}_5$ advantage as

$$Adv_{\mathcal{A}_5}^{Public\ Verifiability} = Pr\left[b = 1\right]. \quad (4)$$

3. A Secure Data Sharing Scheme with Designated Server

In this section, we propose an efficient construction of secure data sharing scheme with designated server.

$(pk_s, sk_s, pk_r, sk_r) \longleftarrow KeyGen(sp)$;
$ch_w \longleftarrow \mathcal{B}(sp)$;
Forge challenge response $R' \longleftarrow \mathcal{A}_5(sp, ch_w, u_w, pk_s, sk_s, pk_r)$;
$b \in \{0,1\} \longleftarrow \mathcal{B}(R', \tau, ch_w)$;
If pass the verification process, then return 1,
else return 0.

Box 5: Game PV $Exp^{PV}_{\mathcal{A}_5}$.

3.1. Our Construction. Our secure data sharing scheme with designated server is described as follows:

A symmetric bilinear pair is $PG = (G_1, G_T, g, p, e)$, where G_1, G_T are cyclic multiplicative groups of prime order p and g is a generator of G_1. Bilinear mapping $e : G_1 \times G_1 \longrightarrow G_T$ satisfies the following properties: bilinearity: for any $\alpha, \beta \in Z_p^*$, $e(g^\alpha, g^\beta) = e(g,g)^{\alpha\beta}$; computability: for any $A, B \in G_1$, we can effectively calculate $e(A, B)$; nondegeneracy: $e(g, g)$ is a generator of G_T.

$\mathsf{SysGen}(1^k)$: this algorithm inputs a security parameter 1^k and generates symmetric bilinear pair $PG = (G_1, G_T, g, p, e)$, in which G_1 and G_T are two cyclic multiplicative groups of prime order p and g is generator of G_1. $H_1 : G_1 \longrightarrow \{0,1\}^d$, $H_2 : \{0,1\}^* \longrightarrow G_1, H_3 : \{0,1\}^* \longrightarrow \{0,1\}^d, H_4 : \{0,1\}^* \longrightarrow \{0,1\}^v, H_5 : \{0,1\}^* \longrightarrow Z_p^*$ are cryptography hash functions. f is a pseudo-random function, $f : \{0,1\}^k \times \{0,1\}^{\log_2^l} \longrightarrow Z_p^*$. $u, \widetilde{u} \in G_1$ are randomly chosen elements. d, v, l are bit length. The algorithm outputs the system parameter

$$sp = \left(PG, H_1, H_2, H_3, H_4, H_5, f, u, \widetilde{u}\right). \tag{5}$$

$\mathsf{KeyGen}_{\mathsf{receiver}}(\mathsf{sp})$: this algorithm inputs system parameter sp, chooses random number $\beta \in Z_p^*$, and outputs a pair of public and secret key (pk_r, sk_r) for the receiver:

$$\begin{aligned} pk_r &= \left(pk_{r1}, pk_{r2}\right) = \left(g^\beta, \widetilde{u}^\beta\right), \\ sk_r &= \beta. \end{aligned} \tag{6}$$

$\mathsf{KeyGen}_{\mathsf{server}}(\mathsf{sp})$: this algorithm inputs system parameter sp, chooses random number $\alpha \in Z_p^*$, and outputs a pair of public and secret key (pk_s, sk_s) for the cloud server:

$$\begin{aligned} pk_s &= \left(pk_{s1}, pk_{s2}\right) = \left(g^\alpha, u^{1/\alpha}\right), \\ sk_s &= \alpha. \end{aligned} \tag{7}$$

$\mathsf{Enc}(\mathsf{sp}, \mathsf{pk_s}, \mathsf{pk_r}, \mathsf{w_i}, \mathsf{m_j})$: this algorithm inputs system parameter sp, the cloud server public key pk_s, the receiver public key pk_r, the keyword w_i, in which $i = 1, 2, \ldots, n$ are the serial number of the keyword, and the message $m_j \in \{0,1\}^d$, in which $j = 1, 2, \ldots, l$ are the message serial number corresponding to the keyword w_i. It chooses random number $r, r' \in Z_p^*$ and outputs the ciphertext:

$$u_{ij} = \left(C_j, C_{w_i}, \sigma\right), \tag{8}$$

where message ciphertext is $C_j = [C_{j,1}, C_{j,2}]$,

$$\begin{aligned} C_{j,1} &= g^r, \\ k &= pk_{r1}^r, \\ C_{j,2} &= H_1(k) \oplus m_j; \end{aligned} \tag{9}$$

and keyword ciphertext is $C_{w_i} = [A, B]$

$$\begin{aligned} A &= pk_{r_1}^{r'}, \\ B &= H_3\left(e\left(H_2\left(w_i\right), pk_{s1}\right)^{r'}\right); \end{aligned} \tag{10}$$

and the binding tag is $\sigma = H_4(k, m_j, C_j, C_{w_i})$.

$\mathsf{Tag}(\mathsf{sp}, \mathsf{pk_s}, \mathsf{pk_r}, \mathsf{w_i}, \mathsf{C_{w_i}}, \mathsf{C_j})$: this algorithm inputs system parameter sp, the cloud server public key pk_s, the receiver public key pk_r, the keyword w_i, the keyword ciphertext C_{w_i}, and the message ciphertext C_j, in which $i = 1, 2, \ldots, n$ are the serial number of the keyword and $j = 1, 2, \ldots, l$ are the message serial number correspond the keyword w_i. It outputs the verification tag τ_{w_i} and releases it to the third party verifier.

$$\tau_{w_i} = \left[\tau_1, \tau_{ij}\right] = \left[pk_{s1}^{H_5(w_i)} pk_{r1}, g^{H_5(C_{w_i} \| C_j)}\right]. \tag{11}$$

$\mathsf{Trapdoor}(\mathsf{sp}, \mathsf{pk_s}, \mathsf{sk_r}, \mathsf{w})$: this algorithm inputs system parameter sp, the cloud server public key pk_s, the receiver secret key sk_r, and the keyword w and outputs the search trapdoor

$$T_w = \left[T_1, T_2\right] = \left[g^{r_1}, H_2(w)^{1/sk_r} \cdot pk_{s_1}^{r_1}\right]. \tag{12}$$

We have the verification trapdoor VK_w:

$$VK_w = pk_{s1}^{H_5(w)} pk_{r1}. \tag{13}$$

$\mathsf{Test}(\mathsf{sp}, \mathsf{sk_s}, \mathsf{T_w}, \mathsf{u})$: this algorithm inputs system parameter sp, the cloud server secret key sk_s, the trapdoor T_w, and the ciphertext u and outputs ciphertext u if

$$B = H_3\left(e\left(A, \frac{T_2}{T_1^{sk_s}}\right)^{sk_s}\right), \tag{14}$$

and $\perp$ otherwise.

$\mathsf{Dec}(\mathsf{sp}, \mathsf{sk_r}, \mathsf{u})$: this algorithm inputs system parameter sp, the receiver secret key sk_r, and the ciphertext u and outputs the message

$$m = C_{j,2} \oplus H_1\left(C_{j,1}^{sk_r}\right), \tag{15}$$

if

$$\sigma = H_4\left(C_1^{sk_r}, m, C_j, C_{w_i}\right) \tag{16}$$

and $\perp$ otherwise.

Public verify$(\mathsf{sp}, \mathsf{VK_w}, \tau, \mathsf{u})$: when there is a dispute between cloud server and receiver, this algorithm inputs system parameter sp, the verification trapdoor VK_w, the verification tag τ, and the ciphertext u and outputs the verification result 1 if satisfying verification condition, and 0 otherwise. The verification process can be divided into three steps:

(i) Challenge query(sp): the third party verifier inputs the system parameter sp and chooses random number $t \in [1, 2^k - 1], s \in Z_p^*$. It keeps the s as a secret value and stores it. It outputs the challenge query ch_w and sends it to cloud server,

$$ch_w =< t, g_2 = g^s > . \tag{17}$$

(ii) GenProof$(\mathsf{ch_w}, \mathsf{sp}, \mathsf{u})$: the server inputs the returned ciphertext u, the system parameter sp, and the challenge query ch_w. It computes $a_j = f_t(j)$ and outputs the challenge response

$$R = \left(g_2\right)^{\sum_{j\in\{1,\ldots,l\}} a_j H_5(C_w \| C_j)} . \tag{18}$$

(iii) Verify$(\mathsf{sp}, \mathsf{VK_w}, \tau_\mathsf{w}, \mathsf{R}, \mathsf{s})$: the third party verifier inputs the system parameter sp, the verification trapdoor VK_w, the verification tag τ_w, the challenge response R, and the secret value s. By the VK_w and $\tau_w = [\tau_1, \tau_{ij}]$, if the $VK_w = \tau_1$, it can get the τ_{ij} and compute

$$\begin{aligned} a_j &= f_t\left(j\right), \\ R' &= \prod_{j\in\{1,\ldots,l\}} \left(\tau_{ij}\right)^{a_j s} . \end{aligned} \tag{19}$$

It outputs result 1, if $R = R'$ and 0 otherwise.

3.2. Proof of Construction. In the following theorems, we will prove that our scheme satisfies keyword ciphertext security and trapdoor indistinguishability security, document ciphertext security, and the public verifiability security in the proposed security model.

We will prove that our scheme is based on 1-BDHI, BDH, DDH, and CDH hard problems. G_1 and G_T in the following problems are groups of prime order p from bilinear paring unless otherwise is specified.

(1-BDHI)[25]. The 1-Bilinear Diffie-Hellman Inversion Problem: given the two tuples (g, g^a), where $g, g^a \in G_1$, all polynomial time algorithms compute the value $e(g, g)^{1/a} \in G_T$ is hard.

(BDH)[25]. The Bilinear Diffie-Hellman Problem: given the four tuple (g, g^a, g^b, g^c), where $g^a, g^b, g^c \in G_1$, all polynomial time algorithms compute the value $e(g, g)^{abc} \in G_T$ is hard.

Theorem 6. *Under 1-BDHI and BDH hard problems, the secure data sharing scheme with designated server satisfies the keyword ciphertext indistinguishability in random oracle model.*

Proof. (1) Suppose there is a cloud server adversary $\mathscr{A}_1$ that can break our scheme in the IND-CKA 1 security model with advantage ε. In order to solve the 1-BDHI hard problem, let us construct a simulator $\mathscr{B}$ with a problem instance (g, g^x) over the cyclic group (G_1, g, p). Our goal is to compute the value $e(g, g)^{1/x}$.

The entire simulation process includes the following phases:

Setup. Let $sp = (PG, H_1, H_2, H_3, H_4, H_5, f, u, \tilde{u})$. The simulator $\mathscr{B}$ chooses random elements $t, t' \in Z_p^*$ and sets the $u = g^t, \tilde{u} = g^{t'}$. It randomly chooses $a \in Z_p^*$, and it sets

$$\begin{aligned} pk_{r1} &= g^{xa}, \\ pk_{r2} &= \tilde{u}^{xa} = g^{t' xa}, \\ sk_r &= xa, \end{aligned} \tag{20}$$

in which sk_r is unknown to simulator $\mathscr{B}$. Simulator $\mathscr{B}$ sends the $pk_r = (pk_{r1}, pk_{r2})$ to the adversary $\mathscr{A}_1$. $\mathscr{A}_1$ randomly chooses $\alpha \in Z_p^*$ and sets

$$\begin{aligned} pk_{s1} &= g^{\alpha}, \\ pk_{s2} &= u^{1/\alpha} = g^{t/\alpha}, \\ sk_s &= \alpha. \end{aligned} \tag{21}$$

Adversary $\mathscr{A}_1$ sends the $pk_s = (pk_{s1}, pk_{s2})$ to the simulator $\mathscr{B}$.

Hash Query

H_1-Query. The adversary $\mathscr{A}_1$ can query $k_i \in G_1$ to H_1. If there exists a $< k_i, X_i >$ in H_1 list, then the simulator $\mathscr{B}$ responds with $H_1(k_i) = X_i$; otherwise, the simulator $\mathscr{B}$ randomly chooses a value X_i, returns to the adversary $\mathscr{A}_1$, and adds the value $< k_i, X_i >$ to H_1 list. The H_1 list is initially empty.

H_2-Query. The adversary $\mathscr{A}_1$ can query w_i to H_2. The simulator maintains a list of tuples $< w_i, v_i, d_i, e_i >$ and the H_2 list is initially empty. If there exists a w_i in H_2 list, then the simulator $\mathscr{B}$ responds with $H_2(w_i) = v_i$; otherwise, the simulator $\mathscr{B}$ generates a random coin $d_i \in \{0, 1\}$, $pr[d_i = 0] = 1/(q_T + 1)$. It randomly chooses $e_i \in Z_p^*$, sets $v_i = g^{e_i}$ if $d_i = 0$, and sets $v_i = g^{xe_i}$ if $d_i = 1$ and adds the value $< w_i, v_i, d_i, e_i >$ to H_2 list.

H_3-Query. The adversary $\mathscr{A}_1$ can query t_i to H_3. If there exists a $< t_i, V_i >$ in H_3 list, then the simulator $\mathscr{B}$ responds with $H_3(t_i) = V_i$; otherwise, the simulator $\mathscr{B}$ randomly chooses a value V_i, returns to the adversary $\mathscr{A}_1$, and adds the value $< t_i, V_i >$ to H_3 list. The H_3 list is initially empty.

H_4-*Query*. The adversary $\mathscr{A}_1$ can query (k_i, m_i, C_j, C_{w_i}) to H_4. If there exists a $((k_i, m_i, C_j, C_{w_i}), \sigma_i)$ in H_4 list, then the simulator $\mathscr{B}$ responds with $H_4(k_i, m_i, C_j, C_{w_i}) = \sigma_i$; otherwise, the simulator $\mathscr{B}$ randomly chooses a value σ_i, returns to the adversary $\mathscr{A}_1$, and adds the value to H_4 list. The H_4 list is initially empty.

Decryption Query. The adversary $\mathscr{A}_1$ can query u to decryption oracle. $u_{ij} = (C_j, C_{w_i}, \sigma)$ is a decryption query. First, the simulator checks the H_4 list, if a tuple $< (k_i, m_i, C_j, C_{w_i}), \sigma_i >$ exists, and satisfies $e(g, k_i) = e(C_{j,1}, pk_{r1})$; the simulator checks H_1 list whether $C_{j,2} = m_i \oplus H_1(k_i)$, if it satisfies the condition, and then returns m_i or otherwise returns $\perp$.

Trapdoor Query. The adversary $\mathscr{A}_1$ can query w_i to trapdoor oracle. First the simulator checks the H_2 list; if $d_i = 0$, the simulation aborts. Otherwise, the simulator computes

$$T_w = (T_1, T_2) = \left(g^{r'}, g^{xe_i/ax} g^{\alpha r'}\right), \tag{22}$$

in which r' is randomly chosen from Z_p^*. Therefore, the simulator completed the trapdoor query and the trapdoor is correct.

Challenge. The adversary $\mathscr{A}_1$ gives two challenge words w_0, w_1 and the message m^* to the simulator $\mathscr{B}$, $w_b \neq w_i, b \in \{0, 1\}$. The simulator returns a ciphertext u_{w_b}, in which $b \in \{0, 1\}$ is randomly chosen. First the simulator checks the H_2 list; if both $d_0 = 1$ and $d_1 = 1$, the simulation aborts. Otherwise, the simulator selects the $d_b = 0$, randomly chooses $r \in Z_p^*, h \in Z_p^*, V^* \in \{0, 1\}^d$, and computes the ciphertext $u_{w_b} = (C_j, C_{w_b}, \sigma)$ as

$$\begin{aligned} C_j &= \left[C_{j,1}, C_{j,2}\right], \\ C_{j,1} &= g^r, \\ k &= pk_{r1}^r, \\ C_{j,2} &= H_1(k) \oplus m^*, \end{aligned} \tag{23}$$

and C_{w_b} as

$$\begin{aligned} C_{w_b} &= [A, B] = \left[\left(g^{xa}\right)^{h/x\alpha} = \left(pk_{s2}^{1/t}\right)^{ha}, V^*\right], \\ \sigma &= H_4\left(k, m^*, C_j, C_{w_b}\right), \\ r' &= \frac{h}{x\alpha}, \end{aligned} \tag{24}$$

in which r' is unknown for simulator $\mathscr{B}$. The ciphertext u_{w_b} is a correct challenge ciphertext.

Trapdoor Query. The adversary $\mathscr{A}_1$ adaptively makes trapdoor query on w_i,$w_i \neq w_0, w_1$. The simulator $\mathscr{B}$ computes the trapdoor as the above trapdoor query.

Decryption Query. The adversary $\mathscr{A}_1$ can query u to decryption oracle similar above decryption query.

Guess. The adversary $\mathscr{A}_1$ outputs bit b' as its guess.

Through the above description, we have completed the simulation process of the scheme and the simulation is correct. Next we will discuss the indistinguishability of the simulation.

When the hash query is not a challenge hash query $e(H_2(w_b), pk_{s1})^{r'} = e(g^{e_b}, g^{\alpha})^{h/x\alpha}$, the response for the decryption query, trapdoor query, and challenge ciphertext are correct. All random numbers in simulation process are random and independent. Random numbers included

$$\begin{aligned} &X_1, X_2, \ldots, X_{q_{H_1}}, v_1, v_2, \ldots, v_{q_{H_2}} \\ &V_1, V_2, \ldots, V_{q_{H_3}}, \sigma_1, \sigma_2, \ldots, \sigma_{q_{H_4}}, a, \alpha, t, t', r, h. \end{aligned} \tag{25}$$

Therefore, the simulation of the scheme is indistinguishable.

When the hash query is not a challenge hash query, the challenge ciphertext is randomness. Therefore, the adversary wins the game with an advantage 0.

As the assumption, from $e(H_2(w_b), pk_{s1})^{r'} = e(g^{\alpha}, g^{e_b})^{h/x\alpha}$, we can find the BDHI problem solution $e(g, g)^{1/x}$ and the finding probability $P_c = 1/q_{H_3}$.

Next we will discuss the successful of the simulation; the simulator does not abort the simulation in trapdoor query and challenge phase. The probability analysis can be seen in the paper [1]. we omit here the probability $P_{successful} = 1/eq_T$. Therefore, the simulator solves the advantage of the 1-BDHI hard problem is

$$\varepsilon_R = \frac{\varepsilon}{eq_T q_{H_3}}. \tag{26}$$

(2) Suppose there is an outside adversary $\mathscr{A}_2$ (including the receiver) that can break our scheme in IND-CKA 2 security model with advantage ε. In order to solve the BDH hard problem, let us construct a simulator $\mathscr{B}$ with a problem instance (g, g^a, g^b, g^c) over the cyclic group (G_1, g, p). Our goal is to compute the value $e(g, g)^{abc}$.

The entire simulation process includes the following phases.

Setup. Let $sp = (PG, H_1, H_2, H_3, H_4, H_5, f, u, \tilde{u})$. The simulator $\mathscr{B}$ randomly chooses $t_1, t_2 \in Z_p^*$ and sets the $h_1 = g^a, h_2 = g^b, h_3 = g^c, u = h_1^{t_1}, \tilde{u} = h_3^{t_2}$, and it randomly chooses $t_3 \in Z_p^*$ and sets

$$\begin{aligned} pk_{s1} &= g^{at_3}, \\ pk_{s2} &= u^{1/at_3} = g^{t_1/t_3}, \\ sk_s &= at_3, \end{aligned} \tag{27}$$

in which sk_s is unknown to the simulator $\mathscr{B}$. The simulator $\mathscr{B}$ sends the $pk_s = (pk_{s1}, pk_{s2})$ to the adversary $\mathscr{A}_2$. The $\mathscr{A}_2$ randomly chooses $\beta \in Z_p^*$ and generates

$$\begin{aligned} pk_{r1} &= g^{\beta}, \\ pk_{r2} &= \tilde{u}^{\beta}, \\ sk_r &= \beta. \end{aligned} \tag{28}$$

The adversary $\mathscr{A}_2$ sends $pk_r = (pk_{r1}, pk_{r2})$ to simulator $\mathscr{B}$.

Hash Query

H_1-*Query*. The adversary $\mathscr{A}_2$ can query $k_i \in G_1$ to H_1. If there exists a $< k_i, X_i >$ in H_1 list, the simulator $\mathscr{B}$ responds with $H_1(k_i) = X_i$. Otherwise, the simulator $\mathscr{B}$ randomly chooses a value X_i and returns to the adversary $\mathscr{A}_2$. It adds the value $< k_i, X_i >$ to H_1 list. The H_1 list is initially empty.

H_2-*Query*. The adversary $\mathscr{A}_2$ can query w_i to H_2. The simulator maintains a list of tuples $< w_i, v_i, e_i >$ and the H_2 list is initially empty. If there exists a w_i in H_2 list, the simulator B responds with $H_2(w_i) = v_i$. Otherwise, the simulator $\mathscr{B}$ randomly chooses $e_i \in Z_p^*$, sets $v_i = g^{be_i}$, and adds the value to H_2 list.

H_3-*Query*. The adversary $\mathscr{A}_2$ can query t_i to H_3. If there exists a t_i in H_3 list, the simulator $\mathscr{B}$ responds with $H_3(t_i) = V_i$. Otherwise, the simulator $\mathscr{B}$ randomly chooses a value $V_i \in G_1$, returns to the adversary $\mathscr{A}_2$, and adds the value to H_3 list. The H_3 list is initially empty.

H_4-*query*. The adversary $\mathscr{A}_2$ can query (k_i, m_i, C_j, C_{w_i}) to H_4. If there exists a $((k_i, m_i, C_j, C_{w_i}), \sigma_i)$ in H_4 list, the simulator $\mathscr{B}$ responds with $H_4(k_i, m_i, C_j, C_{w_i}) = \sigma_i$. Otherwise, the simulator $\mathscr{B}$ randomly chooses a value σ_i, and returns to the adversary $\mathscr{A}_4$ and adds the value to H_4 list. The H_4 list is initially empty.

Decryption Query. The adversary $\mathscr{A}_2$ can query u to decryption oracle. $u_{ij} = (C_j, C_{w_i}, \sigma)$ is a decryption query. First, the simulator needs check the H_4 list, if a tuple $< (k_i, m_i, C_j, C_{w_i}), \sigma_i >$ exist and satisfie $e(g, k_i) = e(C_{j,1}, pk_{r1})$, the simulator checks H_1 list and whether $C_{j,2} = m \oplus H_1(k_i)$; if it satisfies the condition, m returns; otherwise, $\perp$ returns.

Challenge. The adversary $\mathscr{A}_2$ gives two challenge words w_0, w_1 and the message m^* to the simulator $\mathscr{B}$. The simulator randomly chooses $r, t_4 \in Z_p^*, V^* \in \{0,1\}^d$ and returns a ciphertext $u_{w_{b_1}}$, in which $b_1 \in \{0,1\}$ is randomly chosen. The ciphertext $u_{w_{b_1}} = (C_j, C_{w_{b_1}}, \sigma)$ as

$$\begin{aligned} C_j &= \left[C_{j,1}, C_{j,2}\right], \\ C_{j,1} &= g^r, \\ k &= pk_{r1}^r, \\ C_2 &= H_1(k) \oplus m^*, \end{aligned} \tag{29}$$

and $C_{w_{b_1}}$ as

$$\begin{aligned} C_{w_{b_1}} &= [A, B] = \left[pk_{r1}^{r'} = pk_{r2}^{t_4/t_2}, V^*\right], \\ \sigma &= H_4\left(k, m, C_j, C_{w_i}\right), \\ r' &= ct_4, \end{aligned} \tag{30}$$

in which r' is unknown for simulator $\mathscr{B}$. The ciphertext $u_{w_{b_1}}$ is a correct challenge ciphertext.

Decryption Query. The adversary $\mathscr{A}_2$ can query u to decryption oracle similar above decryption query.

Guess. The adversary $\mathscr{A}_2$ outputs bit b_1' as its guess.

Through the above description, we have completed the simulation process of the scheme and the simulation is correct. Next we will discuss the indistinguishability of the simulation.

When the hash query is not a challenge hash query $e(H_2(w_{b_1}), pk_{s1})^{r'} = e(g,g)^{abct_4t_3e_{b_1}}$, the response for the decryption query and challenge ciphertext are correct. All random numbers in simulation process are random and independent. Random numbers includes

$$\begin{aligned} &X_1, X_2, \ldots, X_{q_{H_1}}, v_1, v_2, \ldots, v_{q_{H_2}} \\ &V_1, V_2, \ldots, V_{q_{H_3}}, \sigma_1, \sigma_2, \ldots, \sigma_{q_{H_4}}, a, b, c, r, t_1, t_2, t_3, t_4. \end{aligned} \tag{31}$$

Therefore, the simulation of the scheme is indistinguishable.

When the hash query is not a challenge hash query $e(H_2(w_{b_1}), pk_{s1})^{r'} = e(g,g)^{abct_4t_3e_{b_1}}$, the challenge ciphertext is randomness. Therefore, the adversary wins the game with an advantage 0.

As an assumption, from the challenge hash query $e(H_2(w_{b_1}), pk_{s1})^{r'} = e(g,g)^{abct_4t_3e_{b_1}}$, we can find the BDH problem solution $e(g,g)^{abc}$ and the probability $P_c = 1/q_{H_3}$.

Next we will discuss the successful of the simulation; the simulator does not abort the simulation in challenge phase. Therefore, the simulator solves the advantage of the BDH hard problem is

$$\varepsilon_R = \frac{\varepsilon}{q_{H_3}}. \tag{32}$$

Therefore, Theorem 6 has been proven. □

We will prove that our schemes are trapdoor indistinguishability following DDH hard problem in Theorem 7.

(DDH)[25]. The Diffie-Hellman Hard Problem: given the four tuple (g, g^a, g^b, g^c), where $g^a, g^b, g^c \in G_1$, all polynomial time algorithms decide the value $c \stackrel{?}{=} ab$ is hard.

Theorem 7. *Under DDH hard problem, the secure data sharing scheme with designated server satisfies the trapdoor indistinguishability in standard model, where the security reduction loss is* $L = 1$.

Proof. Suppose there is an outside adversary $\mathscr{A}_3$ that can break our scheme in IND-Trapdoor security model with advantage ε. In order to solve the DDH hard problem, let us construct a simulator $\mathscr{B}$ with a problem instance (g, g^a, g^b, g^c) over the cyclic group (G_1, g, p). Our goal is to decide whether $c = ab$.

The entire simulation process includes the following phases:

Setup. Let $sp = (PG, H_1, H_2, H_3, H_4, H_5, f, u, \tilde{u})$. The simulator $\mathscr{B}$ randomly chooses $t, t', \beta \in Z_p^*$ and sets

$$u = g^{at},$$
$$\widetilde{u} = g^{t'},$$
$$pk_{r1} = g^{\beta}, \tag{33}$$
$$pk_{r2} = \widetilde{u}^{\beta} = g^{t'\beta},$$
$$sk_r = \beta.$$

The simulator $\mathscr{B}$ randomly chooses $l \in Z_p^*$ and generates key pair

$$pk_{s1} = g^{al},$$
$$pk_{s2} = g^{t/l}, \tag{34}$$
$$sk_s = al,$$

in which sk_s is unknown to simulator $\mathscr{B}$. The simulator $\mathscr{B}$ sends $pk_s = (pk_{s1}, pk_{s2})$ and $pk_r = (pk_{r1}, pk_{r2})$ to adversary $\mathscr{A}_3$.

Trapdoor Query. The adversary $\mathscr{A}_3$ can query w_i to trapdoor oracle. The simulator computes

$$T_w = (T_1, T_2) = \left(g^{r'}, H_1(w_i)^{1/\beta} g^{alr'}\right), \tag{35}$$

in which r' is randomly chosen from Z_p^*. Therefore, the simulator completed the trapdoor query and the trapdoor is correct.

Challenge. The adversary $\mathscr{A}_3$ gives two challenge words w_0, w_1 to the simulator $\mathscr{B}$, $w_{b_1} \neq w_i$. $b_1 \in \{0, 1\}$ is randomly chosen. The simulator returns a trapdoor

$$T_{w_{b_1}} = (T_1, T_2) = \left(g^{b/l}, H_2\left(w_{b_1}\right)^{1/\beta} g^{c}\right), \tag{36}$$

in which $r' = b/l$ is unknown to simulator and g^c is a component of the DDH challenge.

When $g^c = g^{ab}$, $T_{w_{b_1}}$ is a valid challenge trapdoor for w_{b_1}.

Trapdoor Query. The adversary $\mathscr{A}_3$ adaptively makes trapdoor query on w_i, $w_i \neq w_0, w_1$. The simulator $\mathscr{B}$ computes trapdoor as the above trapdoor query.

Guess. The adversary $\mathscr{A}_3$ outputs bit b_1' as its guess.

Through the above description, we have completed the simulation process of the scheme and the simulation is correct. Next we will discuss the indistinguishability of the simulation.

When $c = ab$, the response for trapdoor query and challenge trapdoor are correct. All random numbers in simulation process are random and independent. Random numbers included $a, b, c, r', b/l$. So the simulation of the scheme is indistinguishable. The adversary wins the game with a probability of $1/2 + \varepsilon$ as the breaking assumption.

When $c \neq ab$, since the challenge trapdoor is randomness for the adversary $\mathscr{A}_3$, therefore, the adversary $\mathscr{A}_3$ wins the game with a maximum probability of 1/2.

Next we will discuss the successful of the simulation; the simulator does not abort the simulation in trapdoor query and challenge phase. Therefore, the probability $P_{successful} = 1$.

Therefore, the simulator solves the advantage of the HDH hard problem as follows:

$$\varepsilon_R = \left(\frac{1}{2} + \varepsilon - \frac{1}{2}\right) = \varepsilon. \tag{37}$$

□

We will prove that our scheme are IND-CCA secure following CDH hard problem in Theorem 8.

(CDH)[25]. The Computational Diffie-Hellman Problem: given the three tuple (g, g^a, g^b), $g, g^a, g^b \in G_1$, where G_1 is a general cyclic group of prime order p, all polynomial time algorithms compute the value $g^{ab} \in G_1$ is hard.

Theorem 8. *Under CDH hard problem, the secure data sharing scheme with designated server satisfies document ciphertext indistinguishability in random oracle model, where the security reduction loss is* $L = 2^{\nu} q_{H_1}/(2^{\nu} - q_D)$.

Proof. Suppose there is an adversary (including a malicious server) $\mathscr{A}_4$ that can break our secure data sharing scheme with designated server in IND-CCA security model with advantage ε. In order to solve the CDH hard problem, let us construct a simulator $\mathscr{B}$ with a problem instance (g, g^a, g^b) over the cyclic group (G_1, g, p), and our goal is to compute the value g^{ab}.

The entire simulation process includes the following phases.

Setup. Let $sp = (PG, H_1, H_2, H_3, H_4, H_5, f, u, \widetilde{u})$. The simulator $\mathscr{B}$ randomly chooses $t, t' \in Z_p^*$ and sets $u = g^t, \widetilde{u} = g^{t'}$,

$$pk_{r1} = g^{b},$$
$$pk_{r2} = \widetilde{u}^{b} = g^{t'b}, \tag{38}$$
$$sk_r = b,$$

In which sk_r is unknown to simulator $\mathscr{B}$. Simulator $\mathscr{B}$ sends the $pk_r = (pk_{r1}, pk_{r2})$ to the adversary $\mathscr{A}_4$. $\mathscr{A}_4$ randomly chooses $\alpha \in Z_p^*$ and generates

$$pk_{s1} = g^{\alpha},$$
$$pk_{s2} = g^{t/\alpha}, \tag{39}$$
$$sk_s = \alpha,$$

It sends $pk_s = (pk_{s1}, pk_{s2})$ to the simulator $\mathscr{B}$.

Hash Query

H_1-Query. The adversary $\mathscr{A}_4$ can query $k_i \in G_1$ to H_1. If $e(g, k_i) = e(g^a, g^b)$, the simulator aborts the simulation. Otherwise, if there exists a $< k_i, X_i >$ in H_1 list, the simulator

$\mathscr{B}$ responds with $H_1(k_i) = X_i$; otherwise, it randomly chooses a value X_i, returns to the adversary $\mathscr{A}_4$, and adds the value to H_1 list. The H_1 list is initially empty.

H_2-Query. The adversary $\mathscr{A}_4$ can query w_i to H_2. The simulator maintains a list of tuples $< w_i, v_i, e_i >$ and the H_2 list is initially empty. If there exists a w_i in H_2 list, the simulator $\mathscr{B}$ responds with $H_2(w_i) = v_i$. Otherwise, the simulator $\mathscr{B}$ randomly chooses $e_i \in Z_p^*$, sets $v_i = g^{be_i}$, and adds the value to H_2 list.

H_3-Query. The adversary $\mathscr{A}_4$ can query t_i to H_3. If there exists a t_i in H_3 list, the simulator $\mathscr{B}$ responds with $H_3(t_i) = V_i$. Otherwise, the simulator $\mathscr{B}$ randomly chooses a value $V_i \in G_1$, returns to the adversary $\mathscr{A}_4$, and adds the value to H_3 list. The H_3 list is initially empty.

H_4-Query. The adversary $\mathscr{A}_4$ can query (k_i, m_i, C_j, C_{w_i}) to H_4. If $e(g, k_i) = e(g^a, g^b)$, the simulator aborts the simulation. Otherwise, if there exists a (k_i, m_i, C_j, C_{w_i}) in H_4 list, the simulator $\mathscr{B}$ responds with $H_4(k_i, m_i, C_j, C_{w_i}) = \sigma_i$. Otherwise, it randomly chooses a value σ_i, returns to the adversary $\mathscr{A}_4$, and adds the value to H_4 list. The H_4 list is initially empty.

Decryption Query. The adversary $\mathscr{A}_4$ can query u to decryption oracle. $u_{ij} = (C_j, C_{w_i}, \sigma)$ is a decryption query. First, the simulator needs to check the H_4 list; if a tuple $< (k_i, m_i, C_j, C_{w_i}), \sigma_i >$ exists and satisfies $e(g, k) = e(C_{j,1}, pk_{r1})$, the simulator checks the H_1 list and whether $C_{j,2} = m_i \oplus H_1(k_i)$; if it satisfies the condition, then m_i returns; otherwise $\perp$ returns.

Trapdoor Query. The adversary $\mathscr{A}_4$ can query w_i to trapdoor oracle. The simulator sets

$$T_{w_i} = (T_1, T_2) = \left(g^{r_i}, g^{be_i/b} H_6\left(g^{\alpha r_i}\right) = g^{e_i} H_6\left(g^{\alpha r_i}\right)\right). \quad (40)$$

r_i is randomly chosen from Z_p^*. Therefore, the simulator completed the trapdoor query and the trapdoor is correct.

Challenge. The adversary $\mathscr{A}_4$ gives two challenge messages m_0, m_1 and keyword w^* ($w^* \neq m_0, m_1$) to the simulator $\mathscr{B}$. The simulator $\mathscr{B}$ returns a ciphertext $u_{m_{b_1}}$, in which $b_1 \in \{0, 1\}$, $r' \in Z_p^*$ and $Z^* \in \{0, 1\}^d$ are randomly chosen. The ciphertext $u_{m_{b_1}}$ is as follows:

$$\begin{aligned} C_j &= \left[C_{j,1}, C_{j,2}\right], \\ C_{j,1} &= g^a, \\ C_2 &= Z^*, \\ H_1\left(k^*\right) &= Z^* \oplus m_{b_1}, \\ C_{w^*} &= [A, B] \\ &= \left[\left(g^{br'}\right), H_3\left(e\left(H_2\left(w^*\right)^{r'}, pk_{s1}\right)\right)\right]. \end{aligned} \quad (41)$$

The simulator randomly chooses $\sigma^* \in \{0, 1\}^f$, defines

$$\sigma^* = H_4\left(k^*, m_{b_1}, C_j, C_{w^*}\right), \quad (42)$$

and adds the $< (-, m_{b_1}, C_j, C_{w^*}), \sigma^* >$ ($\mathscr{B}$ does not know the value k^*) to H_4 list.

Trapdoor Query. The adversary $\mathscr{A}_4$ adaptively makes trapdoor query on w_i. The simulator $\mathscr{B}$ computes trapdoor as the above trapdoor query.

Decryption Query. The adversary $\mathscr{A}_4$ can query $u \neq u_{m_{b_1}}$ to decryption oracle similar to the above decryption query.

Guess. The adversary $\mathscr{A}_1$ outputs bit b_1' as its guess.

Through the above description, we have completed the simulation process of the scheme and the simulation is correct. Next we will discuss the indistinguishability of the simulation.

When the hash query is not a challenge hash query $k = g^{ab}$, the response for the decryption query, trapdoor query, and challenge ciphertext are correct. All random numbers in simulation process are random and independent. Random numbers include $X_1, X_2, \ldots, X_{q_{H_1}}, v_1, v_2, \ldots, v_{q_{H_2}}, V_1, V_2, \ldots, V_{q_{H_3}}, \sigma_1, \sigma_2, \ldots, \sigma_{q_{H_4}}, a, b, t, t', r_i, r'$. So the simulation of the scheme is indistinguishable.

When the hash query is not a challenge hash query $k = g^{ab}$, the challenge ciphertext is randomness. Therefore, the adversary wins the game with an advantage 0.

As an assumption, from $H_1(k)$, we can find the correct challenge hash query $k = g^{ab}$, $P_c = 1/q_{H_1}$.

Next we will discuss the successful of the simulation, the simulator dose not abort the simulation in trapdoor query and decryption query.

When the value $H_4(k, m, C_j, C_w)$ has been correctly guessed without invoking H_4, the query $< (k, m, C_j, C_w), \sigma >$ to decryption oracle will be aborted. Therefore, the probability

$$P_{successful} = 1 - \frac{q_D}{2^v}, \quad (43)$$

and the q_D is the number decryption oracle queries.

Therefore, the simulator solves the advantage of the CDH hard problem as follows:

$$\varepsilon_R = \frac{\varepsilon}{q_{H_1}}\left(1 - \frac{q_D}{2^v}\right) = \frac{\varepsilon}{2^v q_{H_1}/\left(2^v - q_D\right)}. \quad (44)$$

□

Theorem 9. *Since H_5 is a cryptography hash function, the secure data sharing scheme with designated server is public verifiability secure.*

Proof. Suppose there is a malicious server $\mathscr{A}_5$ that can break our secure data sharing scheme with designated server in public verifiability security model. Let us construct a

Table 1: Computation cost comparison (ms).

	BDOP [1].	BSW [5].	CZLZ [6].	Our.
Keyword Ciph	$2E_1 + h + e = 6.686$	$E_1 + E_2 + h + 2e = 6.218$	$2E_1 + 4E_2 + h + 4e + PM = 9.069$	$E_1 + E_2 + h + e = 5.406$
Trapdoor	$E_1 + h = 4.495$	$E_1 + h = 4.495$	$3E_1 + 3PM = 4.410$	$3E_1 + h + PM = 7.014$
Test	$e = 0.812$	$E_1 + e + PM = 2.282$	$5E_1 + h + 4e + 3PM = 13.532$	$E_1 + E_2 + e = 2.290$

Table 2: Security comparison.

	BDOP [1].	BSW [5].	CZLZ [6].	Our.
Trap Ind	NO	NO	NO	YES
Message Ciph Ind	-	YES	YES	YES
Keyword Ciph Ind	YES	NO	YES	YES
Off-line KGA	NO	NO	NO	YES
SKA	NO	YES	YES	YES
PVS	NO	NO	NO	YES

simulator $\mathcal{B}$ which simulates the scheme. The adversary $\mathcal{A}_5$ can forge a response R for the search keyword w and satisfy the equation $(sp, VK_w, \tau_w, R, s) = 1$. Since the formula $R = R'$ is established, the simulator can compute the equation

$$\sum_{j\in\{1,\ldots,l\}} a_j H_5\left(C_{\widetilde{w}} \parallel \widetilde{C_j}\right) = \prod_{j\in\{1,\ldots,l\}} \left(\tau_{ij}\right)^{a_j}. \tag{45}$$

Therefore, the adversary needs to compute the

$$H_5\left(C_{\widetilde{w}} \parallel \widetilde{C_j}\right) = H_5\left(C_w \parallel C_j\right). \tag{46}$$

Since H_5 is a cryptography hash function, the probability that adversary outputs $C_{\widetilde{w}} \parallel \widetilde{C_j} \neq C_w \parallel C_j$ and makes the equation equal is negligible. □

About the verification tag security, we can also prove the tag security for third party verifier by the security reduction; we omit the details here.

3.3. Performance Analysis. We use Tables 1 and 2 to show two comparisons between our secure data sharing scheme with designated server and previous schemes. In this section, the word abbreviation Trap Ind, Ciph Ind, Offline KGA, PVS, SKA, and keyword Ciph denote trapdoor indistinguishability, ciphertext indistinguishability, offline keyword guessing attacks, public verifiability security, swapping keyword attacks, and keyword ciphertext, respectively. We use e, E_1, E_2, h, PM to denote a pairing operation, an exponentiation operation in G_1, an exponentiation operation in G_2, a hash operation which map a string to an element of cyclic group, and a multiplication in G_1, respectively. We ignore other hash operation and multiplication.

To evaluate the efficiency of our scheme, we implement these operations on a Core(TM) i7-6500U CPU of 2.50GHz 2.60GHz and 4GB RAM (3.89GB is available) running Ubantu 18.04. We use a Type-A pairing elliptic curve and implemented in the PBC library. For these four schemes, we test the running time of keyword ciphertext generation, trapdoor generation, and test algorithms, respectively.

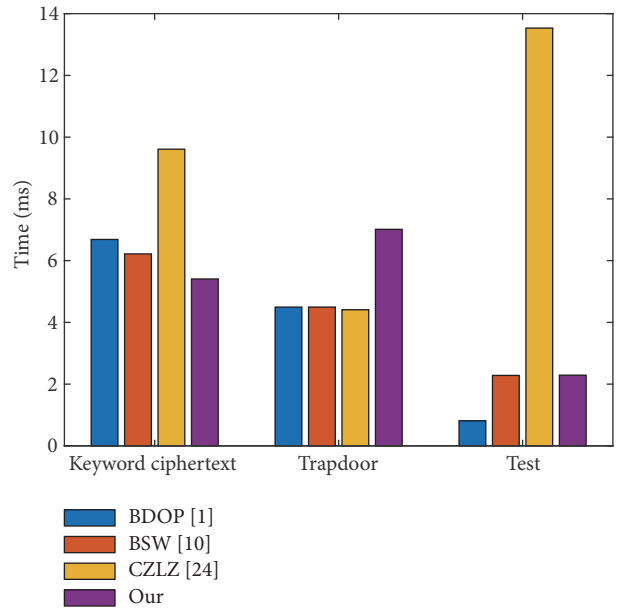

Figure 3: Computation cost of comparison.

We first introduce some basic operation symbols. Every basic operation symbol denotes the running time of an operation in Table 4.

From Figures 3 and 4 and Tables 1 and 4, we found that our scheme is efficient in terms of keyword ciphertext generation algorithm compared to BDOP [1], BSW [5], and CZLZ [6]. Since our scheme reduces some E_1 modular exponentiation computations and e pairing computations, particularly, in CZLZ [6], the scheme requires the most computation cost due to $2E_1$ modular exponentiation computations, $4e$ pairing computations, and $1h$ hash computations per keyword ciphertext generation. In BDOP [1], the scheme requires the most computation cost due to $2E_1$ modular exponentiation computations, $1e$ pairing computations, and $1h$ hash computations per keyword ciphertext generation. In BSW [5], the scheme requires the most computation cost due to $1E_1$ modular exponentiation computations, $2e$ pairing computations, and $1h$ hash computations per keyword ciphertext generation. Our scheme requires the most computation cost due to $1E_1$ modular exponentiations computations, $1e$ pairing computations, and $1h$ hash computations per keyword ciphertext generation; therefore, our scheme is more efficient in terms of keyword ciphertext generation algorithm. As the number of keywords increases from 20 to 80 in Figure 4, we find that our scheme is also efficient.

From Figures 3 and 5 and Tables 1 and 4, we found that our scheme is slightly higher than in terms of trapdoor

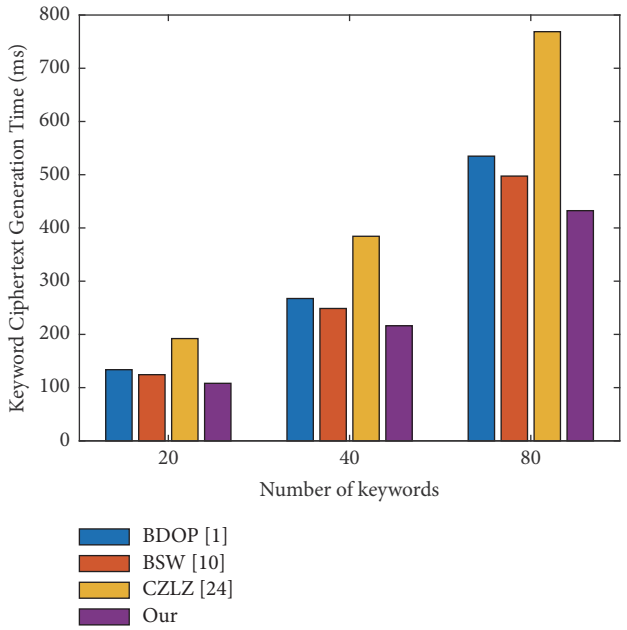

Figure 4: Computation cost of keyword ciphertext generation.

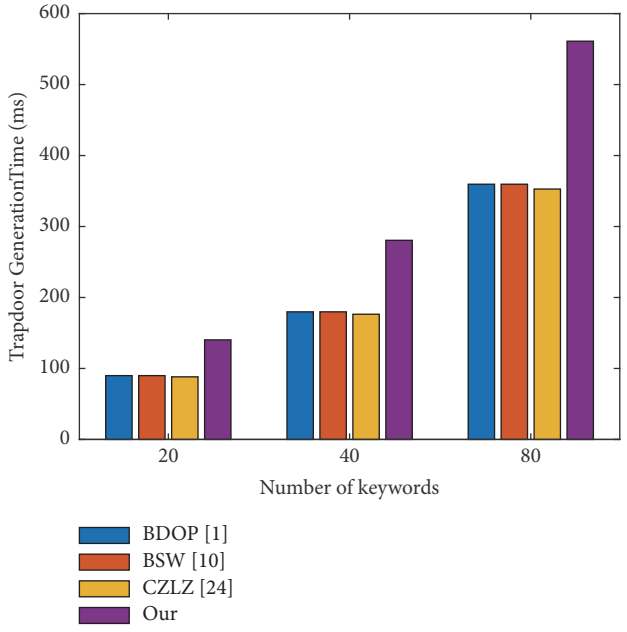

Figure 5: Computation cost of trapdoor generation.

generation algorithm. Since our scheme adds $2E_1$ modular exponentiation computations compared to BDOP [1] and BSW [5], however, it is worth noting that the trapdoor generation in our scheme is slightly higher than those of existing schemes. Since the BDOP [1], BSW [5], and CZLZ [6] suffer from trapdoor security attack, namely, offline KGA, we remove the secure channel between the cloud server and receiver for reducing the construction of secure channel costs. In addition, we can compress the trapdoor computation time from 7.014 ms to 2.876 ms. Since for the trapdoor $T_w = [T_1, T_2] = [g^{r_1}, H_2(w)^{1/sk_r} \cdot pk_{s_1}^{r_1}]$, the $H_2(w)^{1/sk_r}$ is the same value for the same keyword, therefore, we can reduce the $E_1 + h$ computation time when using the same keyword.

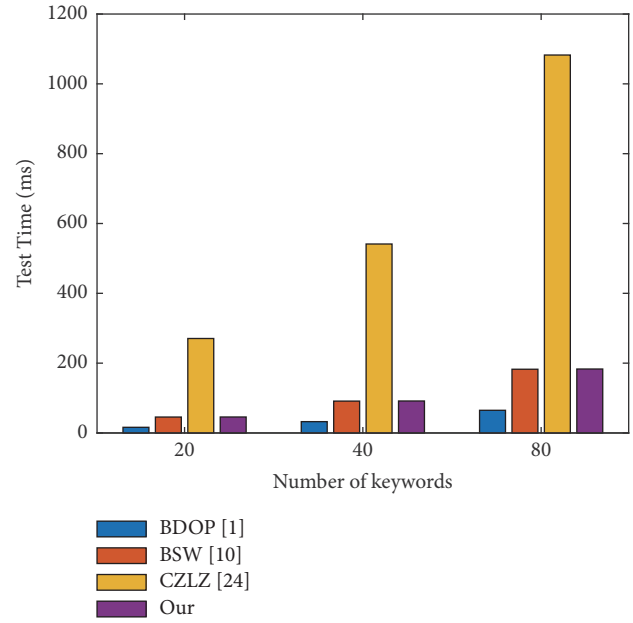

Figure 6: Computation cost of test algorithm.

From Figures 3 and 6 and Tables 1 and 4, we found that our scheme is efficient in terms of test algorithm compared to CZLZ [6], since our scheme reduces $3e$ pairing computations, $4E_1$ modular exponentiation computations, and $1h$ hash computations. However, our scheme is slightly higher than BDOP [1], since we add the $1E_1 + 1E_2$ computation than BDOP [1]. But this is because we remove secure channel. In our scheme, only the designated server can search the keyword via encrypted documents and not need a secure channel between cloud server and receiver. The running test algorithm time of our scheme is almost the same as that of the BSW [5]. In addition, cloud computing has the advantages of unlimited capability in terms of both storage and computation. Therefore, it is acceptable to add a little time during the test phase for reducing the cost of establishing a secure channel.

Furthermore, from Table 2, we find that our scheme also offers stronger security than existing schemes, since our scheme satisfies the trapdoor indistinguishability security, ciphertext indistinguishability security, against offline keyword guessing attacks security, public verifiability security, against swapping keyword attacks security than other schemes. From Table 3, we find that our scheme also offers more functionality than existing schemes, since our scheme offers message and keyword encryption, message decryption, and search result's public verifiability compared to other schemes.

4. Conclusion

We define a secure data sharing scheme with designated server and propose a specific construction. In our framework,

Table 3: Functionality comparison.

	BDOP [1].	BSW [5].	ZH [7].	CZLZ [6].	Our.
Message encryption	NO	NO	YES	YES	YES
Keyword encryption	YES	YES	YES	YES	YES
Message decryption	NO	NO	YES	YES	YES
PV	NO	NO	NO	NO	YES

Table 4: Running time of operations (ms).

e	E_1	E_2	h	PM
0.812	1.379	0.099	3.116	0.091

the scheme not only realizes document encryption/decryption but also achieves the searchable encryption in the cloud environment. We also proved that our scheme has achieved the security of the document indistinguishability against chosen ciphertext attack, keyword indistinguishability against chosen keyword attack, trapdoor indistinguishability, and public verifiability under the proposed security models. The important property of the proposed scheme is that the search results can achieve public verifiability under dishonesty cloud server model, including ciphertexts both documents and keywords. Of course, this scheme can solve the practical scenario PHR problems in our introduction. Although we propose a DSS scheme with designated server that combines PKE scheme with PEKS scheme, we would consider as a major breakthrough to design a DSS scheme in the standard model and optimize the trapdoor size. In addition, as the increase in the number of receivers will degrade the efficiency of system in our scheme, we would consider constructing a DSS scheme in multireceivers scenario.

Data Availability

The data used to support the findings of this study are included in the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by the National Nature Science Foundation of China under Grant no. 61772311 and no. 61272091 and the Open Project of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences no. 2019-ZD-03.

References

[1] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology—EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 506–522, Springer, Berlin, Germany, 2004.

[2] R. Zhang, R. Xue, T. Yu, and L. Liu, "PVSAE: A public verifiable searchable encryption service framework for outsourced encrypted data," in *Proceedings of the 23rd IEEE International Conference on Web Services, ICWS 2016*, pp. 428–435, San Francisco, CA, USA, July 2016.

[3] B. Zhu, J. Sun, J. Qin, and J. Ma, "The public verifiability of public key encryption with keyword search," in *Proceedings of the International Conference on Mobile Networks and Management*, pp. 299–312, Cham, Germany, 2017.

[4] T. Okamoto and D. Pointcheval, "REACT: rapid enhanced-security asymmetric cryptosystem transform," in *Topics in Cryptology — CT-RSA 2001*, vol. 2020 of *Lecture Notes in Computer Science*, pp. 159–174, Springer, Berlin, Heidelberg, 2001.

[5] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proceedings of the International Conference on Computational Science and Applications (ICCSA)*, vol. 5072, pp. 1249–1259, Perugia, Italy, 2008.

[6] Y. Chen, J. Zhang, D. Lin, and Z. Zhang, "Generic constructions of integrated PKE and PEKS," *Designs, Codes and Cryptography*, vol. 78, no. 2, pp. 493–526, 2016.

[7] R. Zhang and H. Imai, "Combining public key encryption with keyword search and public key encryption," *IEICE Transaction on Information and Systems*, vol. E92-D, no. 5, pp. 888–896, 2009.

[8] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P '00)*, pp. 44–55, IEEE, Berkeley, CA, USA, May 2000.

[9] E.-J. Goh, "Secure indexes," *IACR Cryptology ePrint Archive*, p. 216, 2004, https://eprint.iacr.org/2003/216.pdf.

[10] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proceedings of the CCS 2006: 13th ACM Conference on Computer and Communications Security*, pp. 79–88, USA, November 2006.

[11] X. Zhu, H. Dai, X. Yi, G. Yang, and X. Li, "MUSE: an efficient and accurate verifiable privacy-preserving multikeyword text search over encrypted cloud data," *Security and Communication Networks*, vol. 2017, Article ID 1923476, 17 pages, 2017.

[12] M. Abdalla, M. Bellare, D. Catalano et al., "Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions," in *Advances in Cryptology—CRYPTO 2005*, vol. 3621 of *Lecture Notes in Computer Science*, pp. 205–222, 2005.

[13] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *The Journal of Systems and Software*, vol. 83, no. 5, pp. 763–771, 2010.

[14] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Information Sciences*, vol. 238, pp. 221–241, 2013.

[15] N. Andola, S. Prakash, S. Venkatesan, and S. Verma, "Improved secure server-designated public key encryption with keyword search," in *Proceedings of the 2017 IEEE International Conference on Innovative Mechanisms for Industry Applications, ICIMIA 2017*, pp. 1–6, India, February 2017.

[16] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[17] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403-404, pp. 1–14, 2017.

[18] P. Jiang, Y. Mu, F. Guo, and Q. Wen, "Private keyword-search for database systems against insider attacks," *Journal of Computer Science and Technology*, vol. 32, no. 3, pp. 599–617, 2017.

[19] R. Xie, C. He, D. Xie, C. Gao, and X. Zhang, "A secure ciphertext retrieval scheme against insider KGAs for mobile devices in cloud storage," *Security and Communication Networks*, vol. 2018, Article ID 7254305, 7 pages, 2018.

[20] D. Sharma and D. Jinwala, "Multiuser searchable encryption with token freshness verification," *Security and Communication Networks*, vol. 2017, Article ID 6435138, 16 pages, 2017.

[21] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, IEEE, pp. 522–530, Canada, May 2014.

[22] F. Han, J. Qin, H. Zhao, and J. Hu, "A general transformation from KP-ABE to searchable encryption," *Future Generation Computer Systems*, vol. 30, pp. 107–115, 2014.

[23] B. Zhu, J. Sun, J. Qin, and J. Ma, "Fuzzy matching: multi-authority attribute searchable encryption without central authority," *Soft Computing*, vol. 23, no. 2, pp. 527–536, 2019.

[24] J. Baek, R. Safavi-Naini, and W. Susilo, "On the integration of public key data encryption and public key encryption with keyword search," in *Proceedings of the International Conference on Information Security*, pp. 217–232, Samos Island, Greece, 2006.

[25] F. Vercauteren, "Final report on main computational assumptions in cryptography," *ECRPYT*, vol. 11, 2013.