


Article

# A Secure and Efficient Data Sharing and Searching Scheme in Wireless Sensor Networks

Binrui Zhu <sup>1</sup>, Willy Susilo <sup>2</sup> , Jing Qin <sup>1,3,\*</sup>, Fuchun Guo <sup>2</sup>, Zhen Zhao <sup>2</sup> and Jixin Ma <sup>4</sup><sup>1</sup> School of Mathematics, Shandong University, Jinan 250100, China; zhubinrui1509889@163.com<sup>2</sup> School of Computing and Information Technology, University of Wollongong, Wollongong 2522, Australia; wsusilo@uow.edu.au (W.S.); fuchun@uow.edu.au (F.G.); zz343@uowmail.edu.au (Z.Z.)<sup>3</sup> State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China<sup>4</sup> Centre for Computer and Computational Science at School of Computing and Mathematical Sciences, University of Greenwich, London SE10 9LS, UK; j.ma@greenwich.ac.uk

\* Correspondence: qinjing@sdu.edu.cn; Tel.: +86-156-6666-1526

Received: 11 May 2019; Accepted: 3 June 2019; Published: 6 June 2019



**Abstract:** Wireless sensor networks (WSN) generally utilize cloud computing to store and process sensing data in real time, namely, cloud-assisted WSN. However, the cloud-assisted WSN faces new security challenges, particularly outsourced data confidentiality. Data Encryption is a fundamental approach but it limits target data retrieval in massive encrypted data. Public key encryption with keyword search (PEKS) enables a data receiver to retrieve encrypted data containing some specific keyword in cloud-assisted WSN. However, the traditional PEKS schemes suffer from an inherent problem, namely, the keyword guessing attack (KGA). KGA includes off-line KGA and on-line KGA. To date, the existing literature on PEKS cannot simultaneously resist both off-line KGA and on-line KGA performed by an external adversary and an internal adversary. In this work, we propose a secure and efficient data sharing and searching scheme to address the aforementioned problem such that our scheme is secure against both off-line KGA and on-line KGA performed by external and internal adversaries. We would like to stress that our scheme simultaneously achieves document encryption/decryption and keyword search functions. We also prove our scheme achieves keyword security and document security. Furthermore, our scheme is more efficient than previous schemes by eliminating the pairing computation.

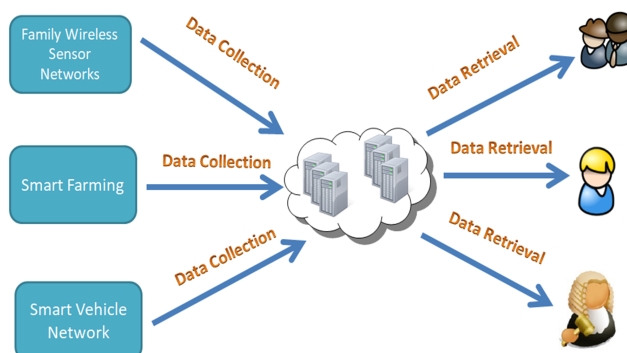
**Keywords:** wireless sensor networks; cloud computing; Internet of Things; public key encryption with keyword search; off-line keyword guessing attack; on-line keyword guessing attack

## 1. Introduction

Wireless sensor networks (WSN) and cloud computing have been widely deployed in daily life. WSN consists of small low-power sensors and lightweight mobile devices connected to the Internet [1,2]. These devices collect and exchange information in a variety of applications. Cloud computing has the advantages of unlimited capability in terms of both storage and computation. WSN is rapidly emerging, which is unprecedentedly driven by the assistance of cloud computing. As an emerging technology, WSN has utilized cloud computing to store and process data to reduce the burden of lightweight mobile devices.

More and more attention has been paid to using WSN technology as a crucial part of the Internet of Things (IoT) in various industries. IoT improves manufacturing efficiency and enables sustainable production [3–7]. As IoT and cloud-assisted WSN applications, enterprises and individuals have utilized cloud storage to complete the data storage and data sharing to reduce the burden of local storage.

As shown in Figure 1, the cloud-assisted WSN typical architecture. In this architecture, the cloud-assisted WSN system has powerful data processing capabilities and storage resources. The sensors implanted in the system collect data information and upload them to the cloud server by using a light mobile device. When the cloud-assisted WSN receives data, it stores and sends the data to relevant industry workers for utilization. In a specific practical scenario, such as a cloud-assisted medical system [8], the medical data documents are confidential to anyone except the patient and the chief physician. Consequently, the stored data should be guaranteed to be secure, since any information disclosure may result in serious consequences. Therefore, security requirements have become a key challenge in cloud-assisted WSN.



**Figure 1.** Functions of a cloud-assisted WSN.

Security issues, such as users' confidence that their data will remain secure with nobody able to modify or observe the contents, will remain the stumbling block that hinders the adoption of cloud-assisted WSN. Generally, users encrypt the data prior to uploading it to the cloud server for protecting data confidentiality. Unfortunately, this approach eliminates the data search services provided by modern search engines, which inevitably makes the effective data search function a challenging research problem. There are two trivial solutions to solve the search problem in encrypted documents. The first one is that the data receiver downloads the encrypted data locally, then decrypts the data and searches for the keyword at the local end. However, this method is impractical since it requires huge communication consumption and occupies a huge local storage space in the WSN. Another way is for the data receiver to send the authorization key to the cloud server which enables it to decrypt the encrypted documents in the cloud and to perform a search operation. However, this approach exposes data privacy to the cloud server and contradicts the original intention of data encryption. Focusing on the aforementioned problem, searchable encryption was proposed [9]. Searchable encryption enables a data receiver to authorize the cloud server to search in encrypted documents and returns the associated encrypted files, where the encrypted documents do not need to be decrypted.

Searchable encryption can be divided into symmetric searchable encryption (SSE) and public key encryption with keyword search (PEKS). In SSE, a shared key is required to achieve a data sharing function. PEKS [10] was proposed to eliminate the shared key in SSE. The general PEKS system includes three participants, that is, data senders, a data receiver and a cloud server. Data senders encrypt the data file and keywords index using the data receiver's public key and then send ciphertexts to the cloud server. The data receiver uses its private key to generate a keyword trapdoor and transmits it to the cloud server. The cloud server uses the trapdoor to match the keyword ciphertext, if the keyword in the ciphertext and the keyword in the trapdoor are equal, it outputs equal; otherwise it outputs not equal.

Unfortunately, the traditional PEKS suffers from an inherent insecurity problem regarding trapdoor privacy. Anyone can use the data receiver's public key to generate the valid keyword ciphertext. If the channel between the data receiver and cloud server is public, then the trapdoor is also

open. If the adversary can execute the test algorithm, then it can verify whether or not the trapdoor and the ciphertext are matched. When they are well matched, the keyword in the trapdoor is equal to the keyword in the ciphertext; otherwise, the adversary can continue to guess another keyword until the correct keyword is found since the keyword space has a much smaller size. This kind of attack is called an off-line keyword guessing attack (off-line KGA), as shown in Figure 2. The off-line KGA is divided into an external adversary’s off-line KGA and an internal server adversary’s off-line KGA, according to which the adversary is an external adversary or an internal server adversary.

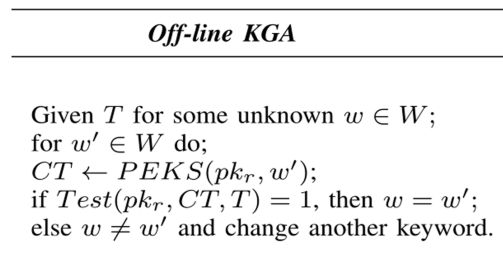


Figure 2. Off-line KGA.

Besides, another inherent insecurity problem regarding trapdoor privacy exists in the traditional PEKS scheme. Since the keyword space has a much smaller size, a malicious data sender (including the external adversary) can generate a data file ciphertext and associated keyword ciphertext by guessing a keyword. If the channel between the data receiver and the cloud server is public, then the trapdoor to locate and return encrypted files is also open. After the cloud server performs the test matching operation, the related encrypted data files are returned. If the returned files have a encrypted data file generated by the malicious sender, the malicious data sender can determine the keyword associated with the encrypted data file, then the keyword in the trapdoor is also known to the malicious data sender. This kind of attack is called an on-line keyword guessing attack (on-line KGA), as shown in Figure 3. The difference between on-line KGA and off-line KGA mainly depends on whether the adversary attacks the scheme through the cloud server.

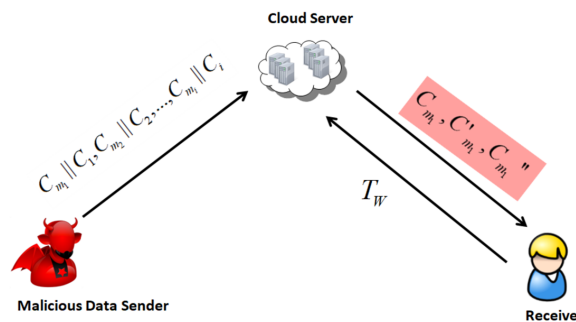


Figure 3. On-line KGA.

For both types of attacks, a trivial solution is that we need a secure channel to share the secret between the data receiver and data senders. A secure channel between cloud server and the data receiver can avoid the off-line KGA initiated by the external adversary and the on-line KGA. But the cost of building a secure channel prevents a Wi-Fi or 4G method from being utilized in the practical application. Moreover, for an internal server adversary, the data receiver and every data sender should share the secret in a secure channel against the off-line KGA initiated by the internal cloud adversary, while this method breaks the asymmetry property of PEKS. Therefore, it is significant and essential to resist both off-line KGA and on-line KGA performed by external and internal adversaries.

Considering a specific scenario: Personal Health Records (PHRs) are confidential documents to anyone except the patient and the chief physician. In order to protect patients' PHR privacy, patients need to encrypt the PHR data prior to uploading it to the cloud server. We want to implement a search function, so a chief physician can search the PHR authorized information. We can use a PEKS scheme to solve the keyword search problem in encrypted PHR. However, the PEKS scheme suffers from an inherent problem, namely, the keyword guessing attack (KGA). In the process of searching, the adversary may obtain the keyword in the trapdoor, which exposes PHR data privacy to the adversary. Therefore, if we can design an efficient and secure data sharing and searching scheme to address the off-line KGA and on-line KGA problem, then data privacy will be guaranteed.

### 1.1. Our Contributions

In this paper, we study how to resist both off-line KGA and on-line KGA performed by external and internal adversaries in PEKS and propose a remedy to these problems. Specifically, our contributions are as follows:

1. We introduce a data sharing and searching (DSS) frame that can effectively resist both off-line KGA and on-line KGA performed by an external adversary and an internal adversary. We also give a specific dual server DSS construction. The security of the scheme can achieve double ciphertext indistinguishability against the on-line KGA and indistinguishability against a chosen keyword attack (IND-CKA). We adopt the dual server method, which divides the cloud server into the forward server and backward server such that any single server cannot complete the test algorithm independently and any single server cannot get the correspondence between trapdoor and keyword ciphertext, therefore, the off-line KGA cannot be conducted successfully.

2. We add data file encryption/decryption to our scheme. In the traditional PEKS scheme, there is no algorithm for data file encryption/decryption. PEKS mainly focuses on the search process and omits the data file encryption/decryption process, which means there is only a keyword encryption algorithm in PEKS and it does not involve a data file encryption/decryption algorithm. However, in the actual application, a data file encryption/decryption is indispensable. The malicious data sender adversary may initiate an on-line KGA by observing the encrypted returned files. We adopt the re-encrypt technique, which the malicious data sender (including backward server) cannot get the correspondence between a trapdoor and encrypted data file, therefore, the on-line KGA cannot be conducted successfully.

3. Our scheme can simultaneously resist both off-line KGA and on-line KGA performed by external and internal adversaries. It does not require a secure channel and keeps the asymmetry property rather than a trivial solution. Compared to the previous schemes, our scheme also improves efficiency by eliminating the pairing computation and offers richer functionality by adding the data file encryption/decryption process.

Technical note: We choose PEKS as the starting point for the design of the scheme. For resisting KGA, we will discuss on-line KGA and off-line KGA. For an external adversary's off-line KGA, the scheme generates a key pair for the cloud server to prevent the external adversary from launching an off-line KGA after eavesdropping the trapdoor through the public channel. What we need to point out here is to generate a key pair for the server it cannot entirely resist an external adversary's off-line KGA. For example, Baek's scheme has a fixed trapdoor. By comparing two bilinear pairs, the adversary can guess a keyword. We also need the trapdoor to satisfy the trapdoor indistinguishability to overcome this external adversary's off-line KGA.

For an internal server adversary's off-line KGA, we can divide the cloud server into two servers, which are the forward server and the backward server. Any single server cannot complete the test algorithm independently. Then, any single server cannot get the correspondence between the trapdoor and the keyword ciphertext, so the off-line KGA cannot be initiated. Therefore, our frame can resist off-line KGA performed by external and internal adversaries.

For on-line KGA, since the attack is initiated by observing the returned data files, we need to consider the data file encryption/decryption. We use the encryption scheme to provide data file encryption/decryption. The malicious data sender observes whether including the returned data file ciphertext is generated by itself to judge the keyword in eavesdropping on the trapdoor. Since the cloud server has strong computing power, we let the forward server perform double encryption for the data file ciphertext. In this way, the generated double ciphertext can satisfy the ciphertext indistinguishability for a malicious data sender, and therefore the malicious data sender adversary cannot initiate on-line KGA.

### 1.2. Related Works

In 2000, Song et al. first proposed an SSE scheme based on a symmetric cryptosystem [9]. Song et al.'s scheme can search any keyword in the ciphertext by word-by-word comparison to complete the keyword search function, therefore, the efficiency is low. Song et al.'s scheme suffers from statistical attacks and it cannot be proven secure. After Song et al.'s scheme, many researchers proposed SSE schemes [11,12]. The symmetric searchable encryption scheme can only be established under the symmetric cryptosystem, therefore, there is a problem of key distribution. In order to solve this problem, Boneh et al. proposed the first PEKS scheme based on the asymmetric cryptosystem in 2004 [10]. Boneh et al.'s scheme is transformed from identity-based encryption (IBE), which replaces the identity in the IBE with the keyword. Boneh et al.'s scheme needs a secure channel between the cloud server and the receiver for uploading the trapdoor. However, the cost of building a secure channel is expensive as is the connection between the receiver and cloud server through an insecure communication channel in IoT environment. In 2005, Abdalla et al. explored the conversion relationship between IBE and PEKS [13]. It is shown that an anonymous IBE scheme could be transformed into a PEKS scheme and it proposed the temporary keyword search scheme. Baek et al. proposed a PEKS scheme to remove the secure channel (dPEKS) [14]. In 2006, Baek et al. proposed a scheme combining a public key encryption (PKE) scheme and PEKS [15]. The scheme achieves the data file encryption/decryption function and keyword search function. Baek et al.'s scheme cannot resist off-line KGA, because the trapdoor in the Baek et al. scheme is fixed, the adversary can test each keyword through a bilinear pair to obtain the keyword in the trapdoor. Rhee et al. improved Baek et al.'s security model in 2009, which allows the adversary to obtain correspondence between the ciphertext and the trapdoor [16].

In 2010, Rhee et al. proposed a new dPEKS scheme [17]. The scheme proposed a new security definition, the trapdoor indistinguishability, and it is a sufficient condition for resisting the external adversary's off-line KGA. In 2013, Fang et al. proposed a scheme that can resist the external adversary's off-line KGA under the standard model [18]. Fang et al.'s scheme is the first dPEKS scheme to achieve the indistinguishability against a chosen keyword ciphertext attack that allows the adversary to initiate test query. Rhee et al.'s two schemes and Fang et al.'s scheme cannot resist internal server's off-line KGA. In 2014, Chen et al. [19] proposed a generalized structure against on-line KGA. Chen et al.'s scheme [19] only satisfies the trapdoor security against on-line KGA and it also suffers from the off-line KGA. In 2016, Chen et al. proposed a two cloud server model [20] and any single server cannot complete the test operation so that it can resist the off-line KGA. However, in Chen et al.'s scheme [20], anyone who can generate a trapdoor and access the test query can create a security problem. It also cannot resist on-line KGA.

In 2016, Chen et al. proposed a joint scheme combining PKE and PEKS [21]. This scheme achieved the IND-CCA security and the indistinguishability against a chosen keyword ciphertext attack security but it could not resist both off-line and on-line KGA. In 2009, Tang et al. proposed a PEKS scheme for resisting off-line KGA [22]. Tang et al.'s method is to share the previously registered keywords between the receiver and every data sender. In 2017, Satio et al. proposed a PEKS scheme of designed-senders [23]. As a designed data sender, it needs to obtain the receiver's authentication. Only the specified data sender can generate valid ciphertext and upload the shared encrypted data to the cloud server; therefore, the internal server adversary cannot generate valid ciphertext and cannot

initiate the off-line KGA. In the same year, Huang et al. [24] and Jiang et al. [25] also used the idea of designed-senders. Only designed-senders can generate valid ciphertext so that it can resist the internal adversary's off-line KGA. In 2018, Wu et al. proposed an off-line KGA scheme against an internal server adversary [26]. It is a method for sharing a secret between the data receiver and every sender. However, all the above five schemes have broken the asymmetry property of PEKS and cannot resist on-line KGA. Zhu et al. proposed a PEKS with a public verifiability scheme [27]. It achieves the public verifiability of the search results, but it cannot resist the internal server's off-line KGA. Han et al. proposed a survey of keyword search schemes in recent years [28]. Many researchers also studied the keyword search problem [29,30].

After we finished our work, we found that Noroozi et al. concurrently presented a generalized PEKS structure against off-line KGA and on-line KGA for an external adversary [31]. It is a method to combine the PEKS with a designated server structure and the technique of re-randomizing ciphertexts. However, it is not enough for the PEKS scheme to resist this external adversary alone. The PEKS scheme still needs to resist an internal server adversary. In our work, we design a PEKS scheme that it simultaneously resists both external adversary and internal server adversary.

Noroozi et al. also considers that designing a PEKS scheme which is secure against off-line KGA and on-line KGA, even performed by the internal server adversary, remains a challenging problem.

We also found that this challenging problem still needs to be addressed. We designed a secure and efficient data sharing and searching (DSS) scheme against both off-line KGA and on-line KGA performed by external and internal adversaries.

### 1.3. Organization

The paper is organized as follows. The scheme definition and security model are described in Section 2. A secure and efficient data sharing and searching scheme against KGA (DSS against KGA) 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 model of the dual server DSS against KGA scheme (Dual server DSS against KGA model) that we proposed is shown in Figure 4. There are four participants in this model including data senders, a receiver, cloud server 1 and cloud server 2. The workflow is as follows:

First of all, data senders encrypt the data file  $M$  using the data receiver's public key  $pk_r$  and encryption algorithm  $Enc$  to form a data file ciphertext  $C_1$ . Data senders also encrypt the corresponding keyword index using two servers' public keys  $pk_{s,1}, pk_{s,2}$ , the receiver's public key  $pk_r$  and the encryption algorithm  $peks$  to form keyword ciphertext  $C_2$ , then sends the ciphertext  $(C_1, C_2)$  to cloud server 1. Secondly, cloud server 1 generates the double ciphertext  $C'_1$  by re-encrypting the data file ciphertext  $C_1$ . Then, the data receiver uses its secret key  $sk_r$  to generate a keyword trapdoor  $T_w$  and transmits it to cloud server 1. Next, cloud server 1 uses the trapdoor  $T_w$  and keyword ciphertext  $C_2$  to compute the transitional ciphertext  $C_T$ , and sends the  $C_T$  to cloud server 2. Afterwards, cloud server 2 outputs the matching result. If the keyword in the ciphertext and the keyword in the trapdoor are equal, cloud server 2 sends the relevant encrypted data file  $C'_1$  to the data receiver. In the final step, to obtain the message  $M$ , the receiver decrypts the data file's double ciphertext  $C'_1$  using its secret key  $sk_r$ .

Although our scheme uses the re-encryption technique, its computational efficiency is almost equal to that of Noroozi et al.'s re-randomizing ciphertexts technique. Of course, the re-encryption technique can also be easily replaced with a re-randomizing ciphertexts technique in our work.

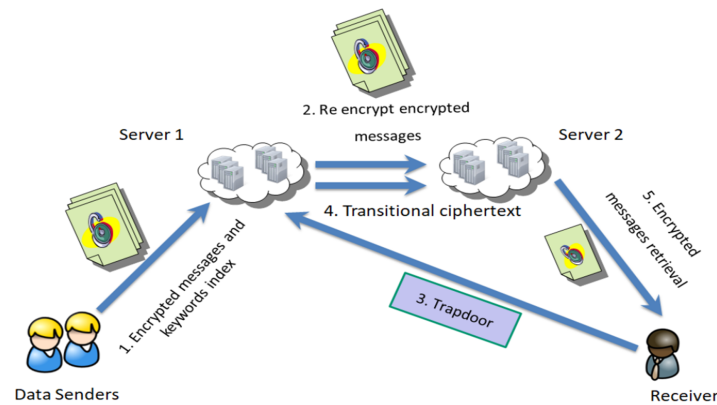


Figure 4. Dual server DSS against KGA model.

## 2.2. Algorithm Definitions

Before defining our algorithms, we define a notations Table 1 for the mathematical symbols in the whole paper.

Table 1. Notations.

Notation	Description
$sp$	System parameter
$pk_{s,1}, sk_{s,1}$	Public/secret key of the cloud server 1
$pk_{s,2}, sk_{s,2}$	Public/secret key of the cloud server 2
$pk_r, sk_r$	Public/secret key of the receiver
$w$	Keyword
$m$	Message
$Enc(m)$	Encryption algorithm Enc for the data $m$
$peks(m)$	Encryption algorithm peks for the keyword $w$
$C_1$	Message ciphertext
$C_2$	Searchable ciphertext for keyword
$C'_1$	Double message ciphertext
$T_w$	Trapdoor for keyword $w$
$C_T$	The transitional ciphertext
$\mathcal{A}$	Adversary
$\mathcal{B}$	Challenger or simulator
$\mathcal{O}(w)$	Trapdoor oracle for the keyword $w$

More specifically, a scheme of DSS against KGA consists of the following algorithms:

- (1)  $sp \leftarrow \text{SysGen}(1^k)$ : on input a security parameter  $k$  and output a system parameter  $sp$ .
- (2)  $\text{KeyGen}(sp)$ :
  - $(pk_{s,1}, sk_{s,1}), (pk_{s,2}, sk_{s,2}) \leftarrow \text{KeyGen}_{\text{server}_{1,2}}(sp)$ : on input a system parameter  $sp$  and output two pairs of public and secret key  $(pk_{s,1}, sk_{s,1}), (pk_{s,2}, sk_{s,2})$  for the cloud server 1 and cloud server 2, separately.
  - $(pk_r, sk_r) \leftarrow \text{KeyGen}_{\text{receiver}}(sp)$ : on input a system parameter  $sp$  and output a pair of public and secret key  $(pk_r, sk_r)$  for the receiver.
- (3)  $(C_1, C_2) \leftarrow \text{PEKS}(sp, pk_{s,1}, pk_{s,2}, pk_r, w, M)$ : on input a system parameter  $sp$ , the cloud server 1 public key  $pk_{s,1}$ , the cloud server 2 public key  $pk_{s,2}$ , the receiver public key  $pk_r$ , the keyword  $w$ , the message  $M$  and output the ciphertext  $C_1 = \text{Enc}(M, pk_r, sp), C_2 = \text{peks}(pk_r, pk_{s,1}, pk_{s,2}, w, sp)$ .

- (4)  $C'_1 \leftarrow \text{ReEnc}(sp, pk_r, C_1)$ : on input a system parameter  $sp$ , the receiver public key  $pk_r$ , the ciphertext  $C_1$ , and output the double ciphertext  $C'_1$ .
- (5)  $T_w \leftarrow \text{Trapdoor}(sp, sk_r, w, pk_{s,1}, pk_{s,2}, pk_r)$ : on input a system parameter  $sp$ , cloud server 1 public key  $pk_{s,1}$ , cloud server 2 public key  $pk_{s,2}$ , the receiver public key  $pk_r$ , the receiver secret key  $sk_r$ , the keyword  $w$ , and output the keyword search trapdoor  $T_w$ .
- (6)  $C'_1$  or  $\perp \leftarrow \text{Test}(sp, T_w, C_2, sk_{s,1}, sk_{s,2})$ : on input a system parameter  $sp$ , the cloud server 1 secret key  $sk_{s,1}$ , the cloud server 2 secret key  $sk_{s,2}$ , the keyword search trapdoor  $T_w$ , the ciphertext  $(C'_1, C_2)$ , and output ciphertext  $C'_1$  if the keyword search trapdoor  $T_w$  matching the ciphertext  $C_2$ , and  $\perp$  otherwise. The matching process as follows:
  - $\text{Test}_1(sp, T_w, C_2, sk_{s,1}) \rightarrow C_T$ : the cloud server 1 inputs the trapdoor  $T_w$ , the ciphertext  $C_2$ , the cloud server 1 secret key  $sk_{s,1}$ , the system parameter  $sp$ , and outputs the transitional ciphertext  $C_T$ .
  - $\text{Test}_2(sp, C_T, sk_{s,2}) \rightarrow C'_1$  or  $\perp$ : the cloud server 2 inputs the system parameter  $sp$ , the transitional ciphertext  $C_T$ , the cloud server 2 secret key  $sk_{s,2}$ . If the transitional ciphertext satisfies the condition, it outputs the double ciphertext  $C'_1$ , and  $\perp$  otherwise.
- (7)  $M \leftarrow \text{Dec}(sp, sk_r, C'_1)$ : on input a system parameter  $sp$ , the receiver secret key  $sk_r$ , the ciphertext  $C'_1$  and output the message  $M$ .

### 2.3. Security Model

We define six security models, including the indistinguishability against a chosen keyword attack (IND-CKA 1) security model for cloud server 1, the IND-CKA 2 security model for cloud server 2, trapdoor indistinguishability against the off-line KGA (IND-Trapdoor 1) security model for cloud server 1, trapdoor indistinguishability against the off-line KGA (IND-Trapdoor 2) security model for cloud server 2, double ciphertext indistinguishability against the on-line KGA (IND-Double ciphertext) security model, transitional ciphertext indistinguishability against chosen keyword attack (IND-CKA 3) security model.

It should be noted that both cloud server 1 and cloud server 2 are “honest but curious” and they will not collude with each other. More specifically, the two servers strictly enforce the testing process of the algorithm but may be curious about the content of the keyword. It should be noted that these models implicitly define the security against external adversaries since the external adversary has less capability than the cloud server.

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

For the IND-CKA 1 security model, as the Table 2, the challenger  $\mathcal{B}$  generates three key pairs  $(pk_{s,1}, sk_{s,1}), (pk_{s,2}, sk_{s,2}), (pk_r, sk_r)$ . It sends public keys  $pk_{s,1}, pk_{s,2}, pk_r$  and secret key  $sk_{s,1}$  to the cloud server 1 adversary  $\mathcal{A}_1$ .  $\mathcal{A}_1$  can access the trapdoor oracle  $\mathcal{O}_1(w)$  to get any keyword trapdoor  $w_i$  and outputs two distinct challenge keywords and a message  $(w_0, w_1, M^*)$ , which  $w_b \neq w_i, b \in \{0, 1\}$ . The challenger  $\mathcal{B}$  generates challenge PEKS ciphertext  $(C_1, C_{2,b})$  of  $(w_b, M^*)$  with a random bit  $b$  and sends it to  $\mathcal{A}_1$ . During the game, the adversary can adaptively continue to query trapdoor oracle  $\mathcal{O}_1(w)$  unless the challenge keywords  $w_0$  and  $w_1$ . Finally, the adversary  $\mathcal{A}_1$  outputs  $b'$  as its guess.

For the IND-CKA 2 security model, as the Table 3, the game is similar to IND-CKA 1. We define security model IND-CKA 2 played between a challenger  $\mathcal{B}$  and adversary  $\mathcal{A}_2$ . We omit the details here. The definition is as follows:

**Definition 1** (IND-CKA). *A scheme of DSS against the KGA is indistinguishable against a chosen keyword attack if no PPT adversaries  $\mathcal{A}_1$  can win game IND-CKA 1 and  $\mathcal{A}_2$  can win game IND-CKA 2 with a non-negligible advantage, where  $\mathcal{B}$  is the challenger,  $\mathcal{A}_1$  is cloud server 1,  $\mathcal{A}_2$  is cloud server 2.*



Table 2. IND-CKA 1.

Game IND-CKA 1 $Exp_{\mathcal{A}_1}^{CKA}[DSS]$
$Kset \leftarrow \phi$ $(pk_{s,1}, sk_{s,1}, pk_{s,2}, sk_{s,2}, pk_r, sk_r) \leftarrow KeyGen(sp);$ $(w_0, w_1, M^*) \leftarrow \mathcal{A}_1^{\mathcal{O}}(sp, (pk_{s,1}, sk_{s,1}), pk_{s,2}, pk_r);$ $(C_1, C_{2,b}) \leftarrow \mathcal{B}(M^*, w_b, pk_{s,1}, pk_{s,2}, pk_r, b \in \{0, 1\});$ $b' \leftarrow \mathcal{A}_1^{\mathcal{O}}(C_1, C_{2,b}, guess);$ if $\{w_0, w_1\} \cap Kset = \phi$ , then return 1, if $b' = b$ ; else return 0. Oracle $\mathcal{O}(w)$ : $Kset = Kset \cup \{w\}, T_w \leftarrow \mathcal{O}(pk_{s,1}, pk_{s,2}, pk_r, sk_r, w);$ return $\{T_w\}$

Table 3. IND-CKA 2.

Game IND-CKA 2 $Exp_{\mathcal{A}_2}^{CKA}[DSS]$
$Kset \leftarrow \phi$ $(pk_{s,1}, sk_{s,1}, pk_{s,2}, sk_{s,2}, pk_r, sk_r) \leftarrow KeyGen(sp);$ $(w_0, w_1, M^*) \leftarrow \mathcal{A}_2^{\mathcal{O}}(sp, (pk_{s,2}, sk_{s,2}), pk_{s,1}, pk_r);$ $(C'_1, C_{2,b}) \leftarrow \mathcal{B}(M^*, w_b, pk_{s,1}, pk_{s,2}, pk_r, b \in \{0, 1\});$ $b' \leftarrow \mathcal{A}_2^{\mathcal{O}}(C'_1, C_{2,b}, guess);$ if $\{w_0, w_1\} \cap Kset = \phi$ , then return 1, if $b' = b$ ; else return 0. Oracle $\mathcal{O}(w)$ : $Kset = Kset \cup \{w\}, T_w \leftarrow \mathcal{O}(pk_{s,1}, pk_{s,2}, pk_r, sk_r, w);$ return $\{T_w\}$

We define  $A_i$  advantage as:

$$Adv_{\mathcal{A}_i}^{IND-CKA} = |\Pr[b = b'] - 1/2|, i \in \{1, 2\}.$$

Next, we define the keyword trapdoor semantic security. Any adversary cannot distinguish the challenge trapdoor, that is to say, the challenge trapdoor does not reveal any information about the keyword. Formally, we define security model IND-Trapdoor 1 and IND-Trapdoor 2 played between a challenger  $\mathcal{B}$  and adversary  $\mathcal{A}_i, i = 3, 4$ .

The IND-Trapdoor 1 and IND-Trapdoor 2 are similar to the IND-CKA 1. The adversary is given the challenge trapdoor instead of the PEKS challenge ciphertext. For the IND-Trapdoor 1 security model, as the Table 4, the challenger  $\mathcal{B}$  generates three key pairs  $(pk_{s,1}, sk_{s,1}), (pk_{s,2}, sk_{s,2}), (pk_r, sk_r)$ . It sends public keys  $pk_{s,1}, pk_{s,2}, pk_r$  and secret key  $sk_{s,1}$  to the cloud server 1 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  $(w_0, w_1)$ , which  $w_b \neq w_i, b \in \{0, 1\}$ . The challenger generates challenge trapdoor  $T_{w_b}$  of  $w_b$  with a random bit  $b$  and sends it to  $\mathcal{A}_3$ . During the game, the adversary can adaptively continue to query trapdoor oracle  $\mathcal{O}_1(w)$  unless the challenge keywords  $w_0$  and  $w_1$ . Finally, the adversary  $\mathcal{A}_3$  outputs  $b'$  as its guess.

For the IND-Trapdoor 2 security model, as the Table 5, the game is similar to IND-Trapdoor 1. We define security model IND-Trapdoor 2 played between a challenger  $\mathcal{B}$  and adversary  $\mathcal{A}_4$ . We omit the details here. The definition is as follows:

**Definition 2 (IND-Trapdoor).** A scheme of DSS against the KGA is trapdoor indistinguishability against off-line KGA if no PPT adversaries  $\mathcal{A}_3$  can win the game IND-Trapdoor 1 and  $\mathcal{A}_4$  can win game IND-Trapdoor 2 with non-negligible advantage, where  $\mathcal{B}$  is the challenger,  $\mathcal{A}_3$  is cloud server 1,  $\mathcal{A}_4$  is cloud server 2.

**Table 4.** IND-Trapdoor 1.

---

*Game IND-Trapdoor 1*  $Exp_{\mathcal{A}_3}^{off-line KGA} [DSS]$

---

$Kset \leftarrow \phi$   
 $(pk_{s,1}, sk_{s,1}, pk_{s,2}, sk_{s,2}, pk_r, sk_r) \leftarrow KeyGen(sp);$   
 $(w_0, w_1) \leftarrow \mathcal{A}_3^{\mathcal{O}}(sp, (pk_{s,1}, sk_{s,1}), pk_{s,2}, pk_r);$   
 $T_b \leftarrow \mathcal{B}(w_b, pk_{s,1}, pk_{s,2}, pk_r, sk_r, b \in \{0, 1\});$   
 $b' \leftarrow \mathcal{A}_3^{\mathcal{O}}(T_b, guess);$   
 if  $\{w_0, w_1\} \cap Kset = \phi$ , then return 1, if  $b' = b$ ;  
 else return 0.  
 Oracle  $\mathcal{O}(w)$ :  
 $Kset = Kset \cup \{w\}, T_w \leftarrow \mathcal{O}(pk_{s,1}, pk_{s,2}, pk_r, sk_r, w);$   
 return  $\{T_w\}$

---

**Table 5.** IND-Trapdoor 2.

---

*Game IND-Trapdoor 2*  $Exp_{\mathcal{A}_4}^{off-line KGA} [DSS]$

---

$Kset \leftarrow \phi$   
 $(pk_{s,1}, sk_{s,1}, pk_{s,2}, sk_{s,2}, pk_r, sk_r) \leftarrow KeyGen(sp);$   
 $(w_0, w_1) \leftarrow \mathcal{A}_4^{\mathcal{O}}(sp, pk_{s,1}, (pk_{s,2}, sk_{s,2}), pk_r);$   
 $T_b \leftarrow \mathcal{B}(w_b, pk_{s,1}, pk_{s,2}, pk_r, sk_r, b \in \{0, 1\});$   
 $b' \leftarrow \mathcal{A}_4^{\mathcal{O}}(T_b, guess);$   
 if  $\{w_0, w_1\} \cap Kset = \phi$ , then return 1, if  $b' = b$ ;  
 else return 0.  
 Oracle  $\mathcal{O}(w)$ :  
 $Kset = Kset \cup \{w\}, T_w \leftarrow \mathcal{O}(pk_{s,1}, pk_{s,2}, pk_r, sk_r, w);$   
 return  $\{T_w\}$

---

We define  $A_i$  advantage as:

$$Adv_{\mathcal{A}_i}^{IND-Trapdoor} = |\Pr[b = b'] - 1/2|, i \in \{3, 4\}.$$

After that, we define the double ciphertext semantic security. Any adversary cannot distinguish the challenge double ciphertext. Formally, we define the IND-Double ciphertext security model, as the Table 6. The IND-Double ciphertext is similar to the IND-CKA 1. The adversary outputs two distinct challenge ciphertext  $(C_{1,0}, C_{1,1})$ . The challenger generates double challenge ciphertext  $C'_{1,b}$  of  $C_{1,b}$  with a random bit  $b$  and sends it to adversary. The adversary is given the challenge double ciphertext instead of the PEKS challenge ciphertext. Finally, the adversary outputs  $b'$  as its guess.

**Definition 3** (IND-Double ciphertext). A scheme of DSS against the KGA is double ciphertext indistinguishability against the on-line KGA if no PPT adversary  $\mathcal{A}_5$  can win the game IND-Double ciphertext with non-negligible advantage, where  $\mathcal{B}$  is the challenger,  $\mathcal{A}_5$  is the malicious data sender (including the cloud server 2).

**Table 6.** IND-Double ciphertext.

---

*Game IND-Double ciphertext*  $Exp_{\mathcal{A}_5}^{Online-KGA} [DSS]$

---

$(pk_{s,1}, sk_{s,1}, pk_{s,2}, sk_{s,2}, pk_r, sk_r) \leftarrow KeyGen(sp);$   
 $(C_{1,0}, C_{1,1}) \leftarrow \mathcal{A}_5(sp, pk_{s,1}, (pk_{s,2}, sk_{s,2}), pk_r);$   
 $C'_{1,b} \leftarrow \mathcal{B}(C_{1,b}, pk_{s,1}, pk_{s,2}, pk_r, b \in \{0, 1\});$   
 $b' \leftarrow \mathcal{A}_5(C'_{1,b}, guess);$   
 Then return 1, if  $b' = b$ ; else return 0.

---

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

$$Adv_{\mathcal{A}_5}^{IND-Double\ ciphertext} = |\Pr[b = b'] - 1/2|.$$

Finally, we define the transitional ciphertext semantic security. Any adversary can not distinguish the challenge transitional ciphertext unless the trapdoor is available. Formally, we define security model IND-CKA 3, as the Table 7. The IND-CKA 3 is similar to the IND-CKA 1. The adversary is given the challenge transitional ciphertext instead of the PEKS challenge ciphertext. We omit the details here.

**Definition 4** (IND-CKA 3). *A scheme of DSS against the KGA is transitional ciphertext indistinguishability against chosen keyword attack if no PPT adversary  $\mathcal{A}_6$  can win the game IND-CKA 3 with non-negligible advantage, where  $\mathcal{B}$  is the challenger and  $\mathcal{A}_6$  is an adversary (including the cloud server 2).*

Table 7. IND-CKA 3.

Game IND-CKA 3 $Exp_{\mathcal{A}_6}^{CKA}[DSS]$
$Kset \leftarrow \phi$ $(pk_{s,1}, sk_{s,1}, pk_{s,2}, sk_{s,2}, pk_r, sk_r) \leftarrow KeyGen(sp);$ $(w_0, w_1) \leftarrow \mathcal{A}_6^{\mathcal{O}}(sp, (pk_{s,2}, sk_{s,2}), pk_{s,1}, pk_r);$ $C_{2,b_1} \leftarrow \mathcal{B}(w_{b_1}, pk_{s,1}, pk_{s,2}, pk_r, b_1 \in \{0, 1\});$ $T_{w_{b_2}} \leftarrow \mathcal{B}(w_{b_2}, pk_{s,1}, pk_{s,2}, pk_r, sk_r, b_2 \in \{0, 1\});$ $C_T \leftarrow \mathcal{B}(C_{2,b_1}, T_{w_{b_2}}, pk_{s,1}, sk_{s,1}, pk_{s,2}, pk_r);$ $(b'_1, b'_2) \leftarrow \mathcal{A}_6^{\mathcal{O}}(C_T, guess);$ if $\{w_0, w_1\} \cap Kset = \phi$ , then return 1, if $(b'_1, b'_2) = (b_1, b_2);$ else return 0. Oracle $\mathcal{O}(w)$ : $Kset = Kset \cup \{w\}, T_w \leftarrow \mathcal{O}(pk_{s,1}, pk_{s,2}, pk_r, sk_r, w);$ return $\{T_w\}$

We define  $\mathcal{A}_6$  advantage as:

$$Adv_{\mathcal{A}_6}^{IND-CKA\ 3} = |\Pr[(b'_1, b'_2) = (b_1, b_2)] - 1/2|.$$

### 3. DSS against the KGA

In this section, we will propose a secure and efficient DSS scheme against the KGA. We use the Hashed Elgama scheme and a free channel PEKS scheme to construct the scheme.

#### 3.1. Our Construction

Our instantiation of the proposed DSS general construction is described as follows: we add the receiver key generation algorithm, data file encryption/decryption algorithm and re-encryption algorithm. Meanwhile, we also eliminate the keyword and trapdoor security problem.

$SystemGen(1^k)$ : This algorithm inputs a security parameter  $1^k$ . It outputs a cyclic multiplicative group  $G_1$  of prime order  $p$  and  $g, g_1, g_2 \in G_1$ , which  $g$  is generator of  $G_1$ . It selects three cryptographic hash functions  $H_1 : \{0, 1\}^* \rightarrow \{0, 1\}^n, H_2 : \{0, 1\}^* \rightarrow G_1, H_3 : \{0, 1\}^* \rightarrow \{0, 1\}^{\log_2^p + n}$ . The algorithm outputs the system parameter

$$sp = (G_1, g, g_1, g_2, H_1, H_2, H_3).$$

$KeyGen(sp)$ :

- $\text{KeyGen}_{\text{server}_{1,2}}(sp)$ : This algorithm inputs a system parameter  $sp$ . It chooses random number  $\alpha_1, \alpha_2, \beta_1, \beta_2 \in Z_p^*$ , and outputs the following  $(pk_{s,1}, sk_{s,1})$  and  $(pk_{s,2}, sk_{s,2})$  as the public/secret key pair of cloud server 1 and that of cloud server 2, separately.

$$(pk_{s,1}, sk_{s,1}) = (g_1^{\alpha_1} g_2^{\alpha_2}, (\alpha_1, \alpha_2)),$$

$$(pk_{s,2}, sk_{s,2}) = (g_1^{\beta_1} g_2^{\beta_2}, (\beta_1, \beta_2)).$$

- $\text{KeyGen}_{\text{receiver}}(sp)$ : This algorithm inputs a system parameter  $sp$ . It chooses random number  $c \in Z_p^*$  and outputs a pair of public and secret key  $(pk_r, sk_r)$  for the receiver,

$$pk_r = g^c, sk_r = c.$$

$\text{PEKS}(sp, pk_{s,1}, pk_{s,2}, pk_r, w, M)$ : This algorithm inputs a system parameter  $sp$ , the cloud server public key  $pk_{s,1}, pk_{s,2}$ , the receiver public key  $pk_r$ , the keyword  $w$ , the message  $M \in \{0, 1\}^n$ , and chooses random number  $r_0, r_1 \in Z_p^*$ . It outputs the message ciphertext  $C_1 = (C_{11}, C_{12})$ , which

$$C_{11} = g^{r_0}, k = pk_r^{r_0}, C_{12} = H_1(k) \oplus M.$$

It also outputs keyword ciphertext

$$C_2 = [A, B, C] = [g_1^{r_1}, g_2^{r_1}, pk_{s,1}^{r_1} \cdot pk_{s,2}^{r_1} \cdot pk_r \cdot H_2(w)].$$

$\text{ReEnc}(sp, pk_r, C_1)$ : This algorithm inputs a system parameter  $sp$ , the receiver public key  $pk_r$ , the message ciphertext  $C_1$ . It chooses random number  $r_2 \in Z_p^*$  and outputs the double message ciphertext  $C'_1 = (C'_{11}, C'_{12})$ , which

$$C'_{11} = g^{r_2}, k = pk_r^{r_2}, C'_{12} = H_3(k) \oplus C_{11} \parallel C_{12}.$$

$\text{Trapdoor}(sp, pk_{s,1}, pk_{s,2}, pk_r, sk_r, w)$ : This algorithm inputs a system parameter  $sp$ , the cloud server public key  $pk_{s,1}, pk_{s,2}$ , the receiver secret key  $sk_r$ , the keyword  $w$ , and chooses random number  $r_3 \in Z_p^*$ . It outputs the keyword search trapdoor  $T_w = [T_1, T_2, T_3]$ ,

$$T_1 = g_1^{sk_r r_3}, T_2 = g_2^{sk_r r_3},$$

$$T_3 = pk_{s,1}^{sk_r r_3} \cdot pk_{s,2}^{sk_r r_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w).$$

$\text{Test}(sp, T_w, C_2, sk_{s,1}, sk_{s,2})$ :

- $\text{Test}_1(sp, T_w, C_2, sk_{s,1}) \rightarrow C_T$ : The cloud server 1 inputs the trapdoor  $T_w$ , the ciphertext  $C_2$ , the cloud server 1 secret key  $sk_{s,1}$ , the system parameter  $sp$ , and chooses random number  $d \in Z_p^*$ . It outputs the transitional ciphertext  $C_T = (A^*, B^*, C^*)$ , where

$$T_w \cdot C_2 = (C_{I,1}, C_{I,2}, C_{I,3}),$$

$$C_{I,1} = T_1 \cdot A, C_{I,2} = T_2 \cdot B, C_{I,3} = T_3 \cdot C,$$

$$A^* = C_{I,1}^d, B^* = C_{I,2}^d, C^* = \left( \frac{C_{I,3}}{C_{I,1}^{\alpha_1} C_{I,2}^{\alpha_2}} \right)^d.$$

- $\text{Test}_2(sp, C'_1, C_T, sk_{s,2}) \rightarrow C'_1$  or  $\perp$ : The cloud server 2 inputs the system parameter  $sp$ , the transitional ciphertext  $C_T$ , the cloud server 2 secret key  $sk_{s,2}$ , and the double ciphertext  $C'_1$ . It outputs  $C'_1$ , if

$$\frac{C^*}{A^* \beta_1 B^* \beta_2} = 1_G,$$

and  $\perp$  otherwise.

$\text{Dec}(sp, sk_r, C'_1)$ : This algorithm inputs a system parameter  $sp$ , the receiver secret key  $sk_r$ , the double message ciphertext  $C'_1$  and outputs the message

$$C_{11} \| C_{12} = C'_{1,2} \oplus H_3(C'_{1,1}^{sk_r}), M = C_{1,2} \oplus H_1(C'_{1,1}^{sk_r}).$$

**Correctness:** When assuming the correctly generated ciphertext  $C_2 = [A, B, C]$  for  $w_i$  with a correct trapdoor  $T_w = (T_1, T_2, T_3)$ . Then we can verify the equation for correctness if  $w_i = w$  as follows:

$$T_w C_2 = (C_{I,1}, C_{I,2}, C_{I,3}), C_{I,1} = g_1^{r_1+cr_3}, C_{I,2} = g_2^{r_1+cr_3},$$

$$C_{I,3} = (g_1^{\alpha_1} g_2^{\alpha_2})^{r_1+cr_3} (g_1^{\beta_1} g_2^{\beta_2})^{r_1+cr_3} H_2(w_i) H_2^{-1}(w).$$

$$C_T = (A^*, B^*, C^*), A^* = g_1^{(r_1+cr_3)d}, B^* = g_2^{(r_1+cr_3)d},$$

$$C^* = ((g_1^{\beta_1} g_2^{\beta_2})^{(r_1+cr_3)} H_2(w_i) H_2^{-1}(w))^d.$$

$$\frac{C^*}{A^* \beta_1 B^* \beta_2} = (H_2(w_i) H_2^{-1}(w))^d = 1_G$$

### 3.2. Proof

In the next theorems, we prove that our scheme satisfies indistinguishability against the chosen keyword attack and trapdoor indistinguishability against the off-line KGA, double ciphertext indistinguishability against the on-line KGA, transitional ciphertext indistinguishability against chosen keyword attack.

To prove our scheme security, we will use the widely accepted security reduction method. The security reduction is that if there is an adversary that can break our scheme, then the adversary can solve the hard mathematical problem. Mathematical hard problems are widely accepted and difficult to solve under existing computing ability. By the proof by contradiction, we can prove that our scheme is secure under the corresponding hard problem. By the security reduction, the scheme's evaluation and validation are guaranteed. Related hard problems can be seen in Reference [32].

#### 3.2.1. Keyword Privacy

We prove that our scheme is secure following the Variant Decisional Diffie-Hellman Problem (Variant DDH) hard problem in Theorem 1 and Theorem 2.

**Variant DDH Hard Problem [32]:** Given the five tuple  $(g, g^a, g^b, g^{ac}, Z)$ ,  $g, g^a, g^b, g^{ac}, Z \in G_1$ , where  $G_1$  is a general cyclic group, all polynomial time algorithms decide the value  $Z \stackrel{?}{=} g^{bc}$  is intractable.

**Theorem 1.** Under Variant DDH hard problem, the DSS scheme satisfies the keyword ciphertext indistinguishability in standard model, where the security reduction loss is 2.

**Proof.** (1) Suppose there is a cloud server 1 named adversary  $\mathcal{A}_1$  that can break our scheme in the IND-CKA 1 security model with advantage  $\epsilon$ . In order to solve the Variant DDH hard problem, let's construct a simulator  $\mathcal{B}$  with a problem instance  $(g_1, g_2, g_1^{\alpha_1}, g_2^{\alpha_2})$  over the cyclic group  $G_1$ . The simulation process is as follows:

**Setup.** Let  $sp = (G_1, g, g_1, g_2, H_1, H_2, H_3)$ . The simulator  $\mathcal{B}$  chooses random elements  $\alpha_1, \alpha_2, \beta_1, \beta_2, c \in \mathbb{Z}_p^*$ , and sets

$$(pk_{s,1}, sk_{s,1}) = (g_1^{\alpha_1} g_2^{\alpha_2}, (\alpha_1, \alpha_2)),$$

$$(pk_{s,2}, sk_{s,2}) = (g_1^{\beta_1} g_2^{\beta_2}, (\beta_1, \beta_2)),$$

$$pk_r = g^c, sk_r = c.$$

The simulator  $\mathcal{B}$  sends the public keys  $pk_{s,2}, pk_r, pk_{s,1}, sk_{s,1}$  to adversary  $\mathcal{A}_1$ .  $\mathcal{B}$  keeps the cloud 2's secret key and receiver's secret key for itself.

**Trapdoor Query.** The adversary  $\mathcal{A}_1$  can query  $w_i$  to trapdoor oracle. The simulator chooses random number  $r_3 \in Z_p^*$  and outputs the keyword search trapdoor

$$T_w = [T_1, T_2, T_3] = [g_1^{cr_3}, g_2^{cr_3}, pk_{s,1}^{cr_3} \cdot pk_{s,2}^{cr_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w)].$$

Therefore, the simulator completed the trapdoor query.

**Challenge.** The adversary  $\mathcal{A}_1$  gives two challenge words  $w_0, w_1$  and the message  $m^*$  to the simulator  $\mathcal{B}$ , which  $w_b \neq w_i, b \in \{0, 1\}$ . The simulator returns a ciphertext  $(C_1, C_{2,b})$ .  $b \in \{0, 1\}$  is randomly chosen. The simulator chooses random number  $r_0 \in Z_p^*$ , and the ciphertext  $(C_1, C_{2,b})$  is outputted as:

$$C_1 = (C_{11}, C_{12}), C_{11} = g^{r_0}, k = pk_r^{r_0}, C_{12} = H_1(k) \oplus m^*,$$

$$C_{2,b} = [g_1^{a_1}, g_2^{a_2}, (g_1^{a_1})^{\alpha_1 + \beta_1} (g_2^{a_2})^{\alpha_2 + \beta_2} g^c H_2(w_b)].$$

Let  $r_1 = a_1$ . If  $a_1 = a_2$ , we have

$$C_{2,b} = [A, B, C] = [g_1^{r_1}, g_2^{r_1}, pk_{s,1}^{r_1} \cdot pk_{s,2}^{r_1} \cdot pk_r \cdot H_2(w_b)].$$

Therefore, the challenge keyword ciphertext is a correct ciphertext.

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

**Guess.** The adversary  $\mathcal{A}_1$  outputs  $b'$  as it's guess.

Through the above description, we have completed the simulation process of the scheme and the simulation is correct, since the responses for the trapdoor query and challenge ciphertext are correct. Next, we will discuss the indistinguishable simulation. Random numbers include

$$\alpha_1, \alpha_2, \beta_1, \beta_2, c, r_3, a_1 = a_2.$$

All random numbers in simulation process are randomness. Therefore, the simulation with  $a_1 = a_2$  is indistinguishable, where the adversary wins the game with a probability of  $1/2 + \frac{\epsilon}{2}$  as the breaking assumption.

When the  $a_1 \neq a_2$ , in the following we show the analysis, the adversary wins the game with a maximum probability of  $1/2$ .

Let  $g_2 = g_1^z$ , the adversary knows

$$z, \alpha_1, \alpha_2, \alpha_1 + z\alpha_2, \beta_1 + z\beta_2, c$$

from the public key. The adversary knows

$$a_1, a_2, a_1(\alpha_1 + \beta_1) + za_2(\alpha_2 + \beta_2) + \log_{g_1}^{H(w_b)}.$$

from the challenge ciphertext.

Therefore, if  $\alpha_1, \alpha_2, \beta_1, \beta_2$  are known to the adversary, the adversary can guess the keyword  $w_b$  correctly; else the the adversary cannot guess the keyword  $w_b$  correctly. Since the adversary knows the  $\alpha_1, \alpha_2$ , but not knows  $\beta_1, \beta_2$ , the adversary has no advantage breaking the ciphertext. Therefore, the adversary wins the game with a probability of  $1/2$  by random guess.

Next, we will discuss the successful of the simulation, the simulator dose not abort the simulation in the trapdoor query and challenge phase. Therefore, the probability of successful simulation is  $P_s = 1$ . Therefore, the advantage of the Variant DDH hard problem is

$$\varepsilon_R = (1/2 + \frac{\varepsilon}{2} - 1/2) = \frac{\varepsilon}{2}.$$

(2) Suppose there is a cloud server 2 named adversary  $\mathcal{A}_2$  that can break our scheme in the IND-CKA 2 security model with advantage  $\varepsilon$ . In order to solve the Variant DDH hard problem, let's construct a simulator  $\mathcal{B}$  with a problem instance  $(g_1, g_2, g_1^{a_1}, g_2^{a_2})$  over the cyclic group  $G_1$ . Simulation process is as follows:

**Setup.** Let  $sp = (G_1, g, g_1, g_2, H_1, H_2, H_3)$ . The simulator  $\mathcal{B}$  chooses random elements  $\alpha_1, \alpha_2, \beta_1, \beta_2, c \in Z_p^*$ , and sets

$$\begin{aligned}(pk_{s,1}, sk_{s,1}) &= (g_1^{\alpha_1} g_2^{\alpha_2}, (\alpha_1, \alpha_2)), \\ (pk_{s,2}, sk_{s,2}) &= (g_1^{\beta_1} g_2^{\beta_2}, (\beta_1, \beta_2)), \\ pk_r &= g^c, sk_r = c.\end{aligned}$$

The simulator  $\mathcal{B}$  sends the public key  $pk_{s,2}, sk_{s,2}, pk_{s,1}, pk_r$  to adversary  $\mathcal{A}_2$ .  $\mathcal{B}$  keeps the cloud 1's secret key and receiver's secret key for itself.

**Trapdoor Query.** The adversary  $\mathcal{A}_2$  can query  $w_i$  to trapdoor oracle. The simulator chooses random number  $r_3 \in Z_p^*$  and outputs the keyword search trapdoor

$$T_w = [T_1, T_2, T_3] = [g_1^{cr_3}, g_2^{cr_3}, pk_{s,1}^{cr_3} \cdot pk_{s,2}^{cr_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w)].$$

Therefore, the simulator completed the trapdoor query.

**Challenge.** The adversary  $\mathcal{A}_2$  gives two challenge words  $w_0, w_1$  and the message  $m^*$  to the simulator  $\mathcal{B}$ , which  $w_b \neq w_i, b \in \{0, 1\}$ . The simulator returns a ciphertext  $(C'_1, C_{2,b})$ .  $b \in \{0, 1\}$  is randomly chosen. The simulator chooses random number  $r_2 \in Z_p^*$ , and the ciphertext  $(C'_1, C_{2,b})$  is outputted as:

$$C'_1 = (C'_{11}, C'_{12}), C'_{11} = g^{r_2}, k = pk_r^{r_2}, C'_{12} = H_3(k) \oplus C_1,$$

which  $C_1$  as the message encryption in the proposed scheme. It also outputs keyword ciphertext

$$C_{2,b} = [g_1^{a_1}, g_2^{a_2}, (g_1^{a_1})^{\alpha_1 + \beta_1} (g_2^{a_2})^{\alpha_2 + \beta_2} g^c H_2(w_b)].$$

Let  $r = a_1$ . If  $a_1 = a_2$ , we have

$$C_{2,b} = [A, B, C] = [g_1^{r_1}, g_2^{r_1}, pk_{s,1}^{r_1} \cdot pk_{s,2}^{r_1} \cdot pk_r \cdot H_2(w_b)].$$

Therefore, the challenge keyword ciphertext is correct.

**Trapdoor Query.** The adversary  $\mathcal{A}_2$  adaptively makes trapdoor query on  $w_i, w_i \neq w_0, w_1$ . The simulator  $\mathcal{B}$  computes trapdoor in the same way as above trapdoor query.

**Guess.** The adversary  $\mathcal{A}_2$  outputs  $b'$  as its guess.

As the entire indistinguishable analysis and probability analysis is similar to the above (1), we omit this process.

Therefore, the simulator solves the advantage of the Variant DDH hard problem

$$\varepsilon_R = (1/2 + \frac{\varepsilon}{2} - 1/2) = \frac{\varepsilon}{2}.$$

The Theorem 1 is proven.  $\square$

Because the cloud server has more powerful attack capabilities than the external adversary, the scheme is also secure to external adversaries (including the receiver) in Theorem 1.

**Theorem 2.** Under Variant DDH hard problem, the DSS scheme satisfies trapdoor indistinguishability against the off-line KGA, where the security reduction loss is 2.

**Proof.** (1) Suppose there is a cloud server 1 named adversary  $\mathcal{A}_3$  that can break our scheme in IND-Trapdoor 1 security model with advantage  $\varepsilon$ . In order to solve the Variant DDH hard problem, let us construct a simulator  $\mathcal{B}$  with a problem instance  $(g_1, g_2, g_1^{a_1}, g_2^{a_2})$  over the cyclic group  $G_1$ . The simulation process is as follows:

**Setup.** Let  $sp = (G_1, g, g_1, g_2, H_1, H_2, H_3)$ . The simulator  $\mathcal{B}$  chooses random elements  $\alpha_1, \alpha_2, \beta_1, \beta_2, c \in Z_p^*$ , and sets

$$(pk_{s,1}, sk_{s,1}) = (g_1^{\alpha_1} g_2^{\alpha_2}, (\alpha_1, \alpha_2)),$$

$$(pk_{s,2}, sk_{s,2}) = (g_1^{\beta_1} g_2^{\beta_2}, (\beta_1, \beta_2)),$$

$$pk_r = g^c, sk_r = c.$$

The simulator  $\mathcal{B}$  sends the public key  $pk_{s,2}, pk_r, pk_{s,1}, sk_{s,1}$  to adversary  $\mathcal{A}_1$ .  $\mathcal{B}$  keeps the cloud 2's secret key and the receiver's secret key for itself.

**Trapdoor Query.** The adversary  $\mathcal{A}_3$  can query  $w_i$  to trapdoor oracle. The simulator chooses random number  $r_3 \in Z_p^*$  and outputs the keyword search trapdoor

$$T_w = [T_1, T_2, T_3] = [g_1^{cr_3}, g_2^{cr_3}, pk_{s,1}^{cr_3} \cdot pk_{s,2}^{cr_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w)].$$

Therefore, the simulator completed the trapdoor query.

**Challenge.** The adversary  $\mathcal{A}_3$  gives two challenge words  $w_0, w_1$  to the simulator  $\mathcal{B}$ , which  $w_b \neq w_i, b \in \{0, 1\}$ . The simulator returns a challenge trapdoor  $T_{w_b}$ .  $b \in \{0, 1\}$  is randomly chosen. The ciphertext  $T_{w_b}$  is outputted as:

$$T_{w_b} = [(g_1^{a_1})^c, (g_2^{a_2})^c, ((g_1^{a_1})^{\alpha_1 + \beta_1} (g_2^{a_2})^{\alpha_2 + \beta_2})^c g^{-c} H_2^{-1}(w_b)].$$

Let  $r_3 = a_1$ . If  $a_1 = a_2$ , we have

$$T_w = [T_1, T_2, T_3] = [g_1^{cr_3}, g_2^{cr_3}, pk_{s,1}^{cr_3} \cdot pk_{s,2}^{cr_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w_b)].$$

Therefore, the challenge keyword trapdoor is a correct trapdoor.

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

**Guess.** The adversary  $\mathcal{A}_3$  outputs  $b'$  as it's guess.

As the entire indistinguishable analysis and probability analysis is similar to the above (1), we omit this process. Therefore, the simulator solving of the advantage of the Variant DDH hard problem is

$$\varepsilon_R = (1/2 + \frac{\varepsilon}{2} - 1/2) = \frac{\varepsilon}{2}.$$

(2) Suppose there is a cloud server 2 named adversary  $\mathcal{A}_4$  that can break our scheme in IND-Trapdoor 2 security model with advantage  $\varepsilon$ . The entire simulation process, solution algorithm and indistinguishable analysis is similar to the above (1), so we omit this process.

Therefore, the simulator solves the advantage of the Variant DDH hard problem

$$\varepsilon_R = (1/2 + \frac{\varepsilon}{2} - 1/2) = \frac{\varepsilon}{2}.$$



Therefore, the Theorem 2 is proven.  $\square$

Because the cloud server has more powerful attack capabilities than the external adversary, the scheme is also secure to external adversaries in Theorem 2.

We will prove that our scheme is secure following computational Diffie-Hellman (CDH) hard problem in Theorem 3.

CDH Hard Problem [15]: 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 intractable.

**Theorem 3.** Under the CDH hard problem, the DSS scheme satisfies double ciphertext indistinguishability against on-line KGA in a random oracle model, where the security reduction loss is  $\frac{1}{q_{H_3}}$ .

**Proof.** Suppose there is an external adversary (including a cloud server 2)  $\mathcal{A}_5$  that can break our scheme in double ciphertext indistinguishability against on-line KGA security model with advantage  $\varepsilon$ . Suppose  $H_3$  as a random oracle, in order to solve the CDH hard problem, let us construct simulator  $\mathcal{B}$  with a problem instance  $(g, g^a, g^b)$  over the cyclic group  $(G_1, g, p)$ . Our goal is to compute the value  $g^{ab}$ . The entire simulation process is as follows:

**Setup.** Let  $sp = (G_1, g, g_1, g_2, H_1, H_2, H_3)$ . The simulator  $\mathcal{B}$  chooses random elements  $\alpha_1, \alpha_2, \beta_1, \beta_2, c \in Z_p^*$ , and sets

$$(pk_{s,1}, sk_{s,1}) = (g_1^{\alpha_1} g_2^{\alpha_2}, (\alpha_1, \alpha_2)),$$

$$(pk_{s,2}, sk_{s,2}) = (g_1^{\beta_1} g_2^{\beta_2}, (\beta_1, \beta_2)),$$

$$pk_r = g^a, sk_r = a.$$

$sk_r$  is unknown to the simulator. The simulator  $\mathcal{B}$  sends the public key  $(pk_{s,2}, sk_{s,2}), pk_{s,1}, pk_r$  to adversary  $\mathcal{A}_5$  and keeps the cloud 1 secret key for itself.

**$H_3$ -query:** The  $H_3$  list is initially empty. The adversary  $\mathcal{A}_5$  can query  $k_i \in G_1$  to  $H_3$ . If there exists a  $(k_i, X_i)$  in  $H_3$  list, then the simulator  $\mathcal{B}$  responds with  $H_3(k_i) = X_i$ ; otherwise, the simulator  $\mathcal{B}$  randomly chooses a value  $X_i \in \{0, 1\}^{\log_2^p + n}$  and sets  $H_1(k_i) = X_i$ . It returns to the adversary  $\mathcal{A}_5$  and adds the value to  $H_3$  list.

**Challenge.** The adversary  $\mathcal{A}_5$  gives two challenge ciphertext  $C_{1,0}, C_{1,1}$  to the simulator  $\mathcal{B}$ . The simulator  $\mathcal{B}$  returns ciphertext  $C'_{1,b_0}$ .  $b_0 \in \{0, 1\}$  is randomly chosen. The ciphertext  $C'_{1,b_0}$  is outputted as:

$$C'_{1,b_0} = [C'_{1,1}, C'_{2,2}], C'_{1,1} = g^b, C'_{2,2} = Z^*, Z^* \in \{0, 1\}^d.$$

Define

$$H_3(g^{ab}) = Z^* \oplus C_{1,b_0},$$

**Guess.** The adversary  $\mathcal{A}_5$  outputs  $b'_0$  as its guess.

$Z^*$  is randomly chosen from  $\{0, 1\}^d$ . When the adversary does not query  $g^{ab}$  to the random oracle, the challenge ciphertext is correct. Through the above description, we have completed the simulation process of the scheme and the simulation is correct. Next we will discuss the indistinguishable simulation. Random numbers include

$$X_1, X_2, \dots, X_{q_{H_3}}, a, b, \alpha_1, \alpha_2, \beta_1, \beta_2, c.$$

Therefore, the simulation of the scheme is indistinguishable.

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

The number of hash query is  $q_{H_3}$ .  $\mathcal{A}_5$  can break our scheme with advantage  $\varepsilon$  as the breaking assumption. Therefore, from the  $H_3(k)$  list, we may find the correct challenge hash query  $g^{ab}$ .

The probability of finding the correct challenge hash query is  $P_c = \frac{1}{q_{H_3}}$ . The simulator does not abort the simulation, therefore, the successful probability of the simulation is  $P_s = 1$ .

The simulator solves the advantage of the CDH hard problem as

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

Therefore, the Theorem 3 is proven.  $\square$

To secure against cloud server 1's on-line KGA, we can let cloud server 2 use a re-encryption technique or a randomizing ciphertexts technique, we omit here the details.

**Theorem 4.** Under Variant DDH hard problem, the DSS scheme satisfies the transitional ciphertext indistinguishability in the standard model, where the security reduction loss is 2.

**Proof.** Suppose there is a cloud server 2 named adversary  $\mathcal{A}_6$  that can break our scheme in IND-CKA 3 security model with advantage  $\varepsilon$ . In order to solve the Variant DDH hard problem, let us construct a simulator  $\mathcal{B}$  with a problem instance  $(g_1, g_2, g_1^{a_1}, g_2^{a_2})$  over the cyclic group  $G_1$ . The simulation process is as follows:

**Setup.** Let  $sp = (G_1, g, g_1, g_2, H_1, H_2, H_3)$ . The simulator  $\mathcal{B}$  chooses random elements  $\alpha_1, \alpha_2, \beta_1, \beta_2, c \in Z_p^*$ , and sets

$$(pk_{s,1}, sk_{s,1}) = (g_1^{\alpha_1} g_2^{\alpha_2}, (\alpha_1, \alpha_2)),$$

$$(pk_{s,2}, sk_{s,2}) = (g_1^{\beta_1} g_2^{\beta_2}, (\beta_1, \beta_2)),$$

$$pk_r = g^c, sk_r = c.$$

The simulator  $\mathcal{B}$  sends the public key  $(pk_{s,2}, sk_{s,2}), pk_{s,1}, pk_r$  to adversary  $\mathcal{A}_6$ .  $\mathcal{B}$  keeps the cloud 1's secret key and receiver's secret key for itself.

**Trapdoor Query.** The adversary  $\mathcal{A}_6$  can query  $w_i$  to trapdoor oracle. The simulator chooses random number  $r'_3 \in Z_p^*$  and outputs the keyword search trapdoor

$$T_w = [T_1, T_2, T_3] = [g_1^{cr'_3}, g_2^{cr'_3}, pk_{s,1}^{cr'_3} \cdot pk_{s,2}^{cr'_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w)].$$

The simulator completed the trapdoor query.

**Challenge.** The adversary  $\mathcal{A}_6$  gives two challenge words  $w_0, w_1$  to the simulator  $\mathcal{B}$ .  $w_{b_1}, w_{b_2} \neq w_i, b_1, b_2 \in \{0, 1\}$ . The simulator generates a ciphertext  $C_{2,b_1}$  and trapdoor  $T_{w_{b_2}}$ .  $b_1, b_2 \in \{0, 1\}$  are randomly chosen. The ciphertext  $C_{2,b_1}$  and the trapdoor  $T_{w_{b_2}}$  as:

$$C_{2,b_1} = [g_1^{a_1}, g_2^{a_2}, (g_1^{a_1})^{\alpha_1 + \beta_1} (g_2^{a_2})^{\alpha_2 + \beta_2} g^c H_2(w_{b_1})],$$

$$T_{w_{b_2}} = [g_1^{cr'_3}, g_2^{cr'_3}, pk_{s,1}^{cr'_3} \cdot pk_{s,2}^{cr'_3} \cdot pk_r^{-1} \cdot H_2^{-1}(w_{b_2})].$$

Let  $r_1 = a_1$ . If  $a_1 = a_2$ , we have

$$C_{2,b_1} = [A, B, C] = [g_1^{r_1}, g_2^{r_1}, pk_{s,1}^{r_1} \cdot pk_{s,2}^{r_1} \cdot pk_r \cdot H_2(w_{b_1})].$$

Therefore, the transitional ciphertext is  $C_{T^*} = (A^*, B^*, C^*)$ , where

$$T_{w_{b_2}} C_{2,b_1} = (C_{I,1}, C_{I,2}, C_{I,3}), A^* = C_{I,1}^d, B^* = C_{I,2}^d,$$

$$C^* = ((g_1^{(a_1 + cr_3)\beta_1} g_2^{(a_2 + cr_3)\beta_2}) H_2(w_{b_1}) H_2^{-1}(w_{b_2}))^d.$$

Therefore, the transitional ciphertext is a correct ciphertext.

**Trapdoor Query.** The adversary  $\mathcal{A}_6$  adaptively makes trapdoor query on  $w_i, w_i \neq w_0, w_1$ . The simulator  $\mathcal{B}$  computes the trapdoor in the same way as above trapdoor query.

**Guess.** The adversary  $\mathcal{A}_6$  outputs  $(b'_1, b'_2)$  as it's guess.

When  $a_1 = a_2$ , the indistinguishable analysis is similar to Theorem 1, we omit this process.

When the  $a_1 \neq a_2$ , in the following we show the analysis, the adversary wins the game with probability of  $1/2$ .

Let  $g_2 = g_1^z$ , the adversary knows  $z, \beta_1, \beta_2, \alpha_1 + z\alpha_2, \beta_1 + z\beta_2, c$  from the public key. The adversary knows

$$(a_1 + cr_3)d, (a_2 + cr_3)d, \\ (a_1 + cr_3)d\beta_1 + z(a_2 + cr_3)d\beta_2 + \log_{g_1}^{(H(w_{b_1})H_2^{-1}(w_{b_2}))^d}.$$

from the challenge ciphertext.

Therefore, if the  $d$  is known, the adversary will guess keywords  $(w_{b_1}, w_{b_2})$  correctly; else the

$$(a_1 + cr_3)d\beta_1 + z(a_2 + cr_3)d\beta_2 + \log_{g_1}^{(H(w_{b_1})H_2^{-1}(w_{b_2}))^d}$$

hides the  $(w_{b_1}, w_{b_2})$ . Since the adversary does not know the  $d$ , it also has no advantage in breaking the ciphertext.

Next, the successful simulation probability is  $P_s = 1$ . The simulator solves the advantage of the Variant DDH hard problem as

$$\epsilon_R = (1/2 + \frac{\epsilon}{2} - 1/2) = \frac{\epsilon}{2}.$$

Therefore, the Theorem 4 is proven.  $\square$

### 3.2.2. Message Privacy

Regarding the security of the message, the proof is similar to Theorem 3 and is based on the CDH hard problem in the random oracle model. Since it is too similar, we omit the proof here.

### 3.3. Analysis and Comparisons

We use Tables 8 and 9 to show two comparisons between our scheme and previous schemes. In this section, the word abbreviation Trap Ind, MCiph Ind, KCiph Ind, In-off-line KGA, Ex-off-line KGA, on-line KGA, MCiph, KCiph to denote trapdoor indistinguishability, message ciphertext indistinguishability, keyword ciphertext indistinguishability, off-line keyword guessing attack for internal attacker, off-line keyword guessing attack for external attacker, on-line keyword guessing attack, message ciphertext, keyword ciphertext. We use  $e, E_1, E'_1, E'_2, h, I, PM$  to denote a pairing operation, an exponentiation operation in cyclic multiplicative group  $G_1$ , an exponentiation operation in  $G'_1$  from paring, an exponentiation operation in  $G'_T$  from paring, a hash operation maps a string to an element of cyclic group, an inverse operation, a multiplication in  $G'_1$  from paring. We ignore other hash operations and multiplication.

**Table 8.** Computation comparison.

	BCOP [10]	BSW [14]	RPSL [17]	Our
MCiph	-	-	-	$2E_1$
KCiph	$2E'_1 + h + e$	$E'_1 + E'_2 + h + 2e$	$2E'_1 + h + e$	$4E_1 + h$
ReEnc	-	-	-	$2E_1$
Trapdoor	$E'_1 + h$	$E'_1 + h$	$3E'_1 + 2h + I + PM$	$4E_1 + h + 2I$
Test	$e$	$E'_1 + e + PM$	$2E'_1 + h + e + PM$	$7E_1$
Dec	-	-	-	$2E_1$

Table 9. Security comparison.

	BCOP [10]	BSW [14]	RPSL [17]	Our
Trap Ind	NO	NO	YES	YES
MCiph Ind	-	-	-	YES
KCiph Ind	YES	YES	YES	YES
In-off-line KGA	NO	NO	NO	YES
Ex-off-line KGA	NO	NO	YES	YES
on-line KGA	NO	NO	NO	YES

To evaluate the efficiency of our scheme, we implemented these schemes on a Core(TM) i7-6500U CPU at 2.50GHz and 4GB RAM (3.89GB is available) running Ubuntu 18.04. We used a Type-A pairing elliptic curve and implemented in the PBC library. For these four schemes, we tested the running time of keyword ciphertext generation, trapdoor generation and test algorithms, respectively. The comparison results are shown in Figures 5–7. From these three figures, we found that our scheme is the most efficient in terms of keyword ciphertext generation and trapdoor generation algorithms. Although our scheme’s test algorithm is slightly less computationally efficient than BCOP [10] scheme. However, in comparison with other PEKS schemes, our efficiency remains high by eliminating the pairing computation and exponentiation operation in  $G_1'$ . Furthermore, our scheme also offers a stronger security guarantee for keyword security.

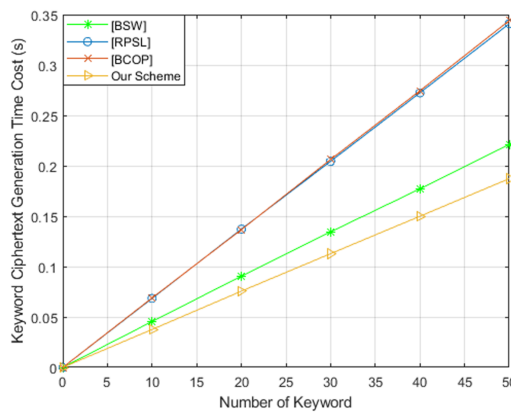


Figure 5. Computation cost of keyword ciphertext generation.

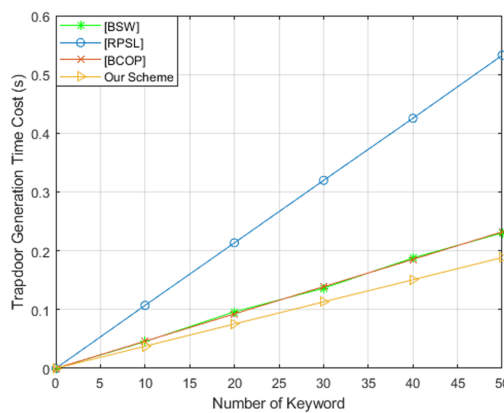


Figure 6. Computation cost of trapdoor generation.

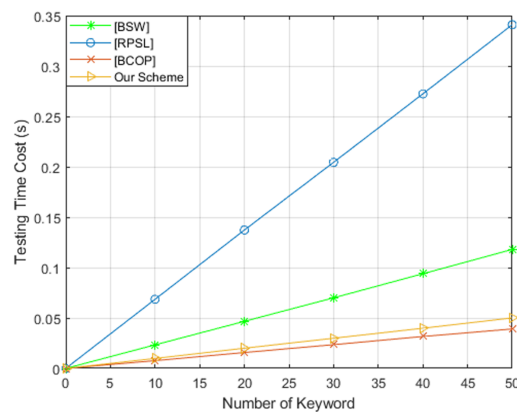


Figure 7. Computation cost of test algorithm.

### 3.4. Research Method

In our paper, we researched the trapdoor security problem in a WSN environment in the following way, which is motivation  $\Rightarrow$  application scenario  $\Rightarrow$  technical route  $\Rightarrow$  frame architecture  $\Rightarrow$  security model  $\Rightarrow$  concrete construction  $\Rightarrow$  security reduction  $\Rightarrow$  efficiency analysis and comparisons.

## 4. Conclusions

The combination of cloud computing and WSN provides a promising solution to handle massive data. Data security requirements have become a key challenge in cloud-assisted WSN. To address limitations inherent in data security problems, in this paper, we defined a secure and efficient DSS scheme that can resist both off-line KGA and on-line KGA performed by external adversary and internal adversary, and we proposed a specific construction. This construction can simultaneously resist both on-line KGA and off-line KGA in cloud-assisted WSN. Our scheme not only realizes the keyword search function in the cloud but also implements the data files encryption/decryption function. The performance analysis shows the computation overhead at lightweight mobile devices is significantly reduced. We also formally proved that our schemes are provably secure.

**Author Contributions:** Conceptualization, B.Z.; methodology, B.Z., W.S., J.Q., F.G.; formal analysis, B.Z.; writing—original draft preparation, B.Z.; writing—review and editing, Z.Z., W.S., J.M.; supervision, J.Q.

**Funding:** This work is supported by the National Nature Science Foundation of China under Grant No: 61772311, 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.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Bista, R.; Chang, J.W. Privacy-Preserving Data Aggregation Protocols for Wireless Sensor Networks: A Survey. *Sensors* **2010**, *10*, 4577–4601. [[CrossRef](#)]
2. Zhang, P.; Ma, J. Channel Characteristic Aware Privacy Protection Mechanism in WBAN. *Sensors* **2018**, *18*, 2403. [[CrossRef](#)] [[PubMed](#)]
3. Sisinni, E.; Saifullah, A.; Han, S.; Jennehag, U.; Gidlund, M. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4724–4734. [[CrossRef](#)]
4. Pease, S.G.; Trueman, R.; Davies, C.; Grosberg, J.; Yau, K.H.; Kaur, N.; Conway, P.; West, A. An intelligent real-time cyber-physical toolset for energy and process prediction and optimisation in the future industrial Internet of Things. *Future Gener. Comput. Syst.* **2018**, *79*, 815–829. [[CrossRef](#)]
5. Jung, H.; Lee, I.H. Secrecy Performance Analysis of Analog Cooperative Beamforming in Three-Dimensional Gaussian Distributed Wireless Sensor Networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 1860–1873. [[CrossRef](#)]
6. Xie, H.; Yan, Z.; Yao, Z.; Atiquzzaman, M. Data Collection for Security Measurement in Wireless Sensor Networks: A Survey. *IEEE Internet Things J.* **2019**, *6*, 2205–2224. [[CrossRef](#)]

7. Yaqoob, I.; Hashem, I.A.T.; Ahmed, A.; Kazmi, S.A.; Hong, C.S. Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges. *Future Gener. Comput. Syst.* **2019**, *92*, 265–275. [[CrossRef](#)]
8. Tan, H.; Chung, I. A Secure and Efficient Group Key Management Protocol with Cooperative Sensor Association in WBANs. *Sensors* **2018**, *18*, 3930. [[CrossRef](#)]
9. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, S&P 2000, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55.
10. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In Proceedings of the Advances in Cryptology—EUROCRYPT 2004, Interlaken, Switzerland, 2–6 May 2004; pp. 506–522.
11. Curtmola, R.; Garay, J.; Kamara, S.; Ostrovsky, R. Searchable symmetric encryption: Improved definitions and efficient constructions. *J. Comput. Secur.* **2011**, *19*, 895–934. [[CrossRef](#)]
12. Chang, Y.C.; Mitzenmacher, M. Privacy Preserving Keyword Searches on Remote Encrypted Data. In Proceedings of the Applied Cryptography and Network Security, New York, NY, USA, 7–10 June 2005; pp. 442–455.
13. Abdalla, M.; Bellare, M.; Catalano, D.; Kiltz, E.; Kohno, T.; Lange, T.; Malone-Lee, J.; Neven, G.; Paillier, P.; Shi, H. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and Extensions. In Proceedings of the Advances in Cryptology—CRYPTO 2005, Santa Barbara, CA, USA, 14–18 August 2005; pp. 205–222.
14. Baek, J.; Safavi-Naini, R.; Susilo, W. Public key encryption with keyword search revisited. In Proceedings of the International conference on Computational Science and Its Applications, Perugia, Italy, 30 June–3 July 2008; pp. 1249–1259.
15. Baek, J.; Safavi-Naini, R.; Susilo, W. On the Integration of Public Key Data Encryption and Public Key Encryption with Keyword Search. In Proceedings of the Information Security, Samos Island, Greece, 30 August–2 September 2006; pp. 217–232.
16. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Improved searchable public key encryption with designated tester. In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, Sydney, Australia, 10–12 March 2009; pp. 376–379.
17. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J. Syst. Softw.* **2010**, *83*, 763–771. [[CrossRef](#)]
18. Fang, L.; Susilo, W.; Ge, C.; Wang, J. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. *Inf. Sci.* **2013**, *238*, 221–241. [[CrossRef](#)]
19. Chen, Y.C. SPEKS: Secure server-designation public key encryption with keyword search against keyword guessing attacks. *Comput. J.* **2014**, *58*, 922–933. [[CrossRef](#)]
20. Chen, R.; Mu, Y.; Yang, G.; Guo, F.; Wang, X. Dual-server public-key encryption with keyword search for secure cloud storage. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 789–798. [[CrossRef](#)]
21. Chen, Y.; Zhang, J.; Lin, D.; Zhang, Z. Generic constructions of integrated PKE and PEKS. *Des. Codes Cryptogr.* **2016**, *78*, 493–526. [[CrossRef](#)]
22. Tang, Q.; Chen, L. Public-key encryption with registered keyword search. In Proceedings of the European Public Key Infrastructure Workshop, Pisa, Italy, 10–11 September 2009; pp. 163–178.
23. Saito, T.; Nakanishi, T. Designated-Senders Public-Key Searchable Encryption Secure against Keyword Guessing Attacks. In Proceedings of the 2017 Fifth International Symposium on Computing and Networking (CANDAR), Aomori, Japan, 19–22 November 2017; pp. 496–502.
24. Huang, Q.; Li, H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. *Inf. Sci.* **2017**, *403*, 1–14. [[CrossRef](#)]
25. Jiang, P.; Mu, Y.; Guo, F.; Wen, Q.Y. Private Keyword-Search for Database Systems Against Insider Attacks. *J. Comput. Sci. Technol.* **2017**, *32*, 599–617. [[CrossRef](#)]
26. Wu, L.; Chen, B.; Zeadally, S.; He, D. An efficient and secure searchable public key encryption scheme with privacy protection for cloud storage. *Soft Comput.* **2018**, *22*, 7685–7696. [[CrossRef](#)]
27. Zhu, B.; Sun, J.; Qin, J.; Ma, J. The Public Verifiability of Public Key Encryption with Keyword Search. In Proceedings of the International Conference on Mobile Networks and Management, Melbourne, Australia, 13–15 December 2017; pp. 299–312.
28. Han, F.; Qin, J.; Hu, J. Secure searches in the cloud: A survey. *Future Gener. Comput. Syst.* **2016**, *62*, 66–75. [[CrossRef](#)]

29. Wu, A.; Zheng, D.; Zhang, Y.; Yang, M. Hidden Policy Attribute-Based Data Sharing with Direct Revocation and Keyword Search in Cloud Computing. *Sensors* **2018**, *18*, 2158. [[CrossRef](#)] [[PubMed](#)]
30. Guo, Y.; Liu, F.; Cai, Z.; Xiao, N.; Zhao, Z. Edge-Based Efficient Search over Encrypted Data Mobile Cloud Storage. *Sensors* **2018**, *18*, 1189. [[CrossRef](#)]
31. Noroozi, M.; Eslami, Z. Public-key encryption with keyword search: A generic construction secure against online and offline keyword guessing attacks. *J. Ambient Intell. Humaniz. Comput.* **2019**. [[CrossRef](#)]
32. Cramer, R.; Shoup, V. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 23–27 August 1998; pp. 13–25.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).