Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8850520, 11 pages
https://doi.org/10.1155/2020/8850520

Research Article

An Efficient Pairing-Free Certificateless Searchable Public Key Encryption for Cloud-Based IIoT

Mimi Ma,[1,2,3] Min Luo,[4] Shuqin Fan,[1] and Dengguo Feng[1]

[1]*State Key Laboratory of Cryptology, Beijing, China*
[2]*Key Laboratory of Grain Information Processing and Control (Henan University of Technology), Ministry of Education, Zhengzhou, China*
[3]*College of Information Science and Engineering, Henan University of Technology, Zhengzhou, China*
[4]*School of Cyber Science and Engineering, Wuhan University, Wuhan, China*

Correspondence should be addressed to Min Luo; mluo@whu.edu.cn

Received 25 June 2020; Revised 6 August 2020; Accepted 2 December 2020; Published 21 December 2020

Academic Editor: Ximeng Liu

The Industrial Internet of Things (IIoT), as a special form of Internet of Things (IoT), has great potential in realizing intelligent transformation and industrial resource utilization. However, there are security and privacy concerns about industrial data, which is shared on an open channel via sensor devices. To address these issues, many searchable encryption schemes have been presented to provide both data privacy-protection and data searchability. However, due to the use of expensive pairing operations, most previous schemes were inefficient. Recently, a certificateless searchable public-key encryption (CLSPE) scheme was designed by Lu et al. to remove the pairing operation. Unfortunately, we find that Lu et al.'s scheme is vulnerable to user impersonation attacks. To enhance the security, a new pairing-free dual-server CLSPE (DS-CLSPE) scheme for cloud-based IIoT deployment is designed in this paper. In addition, we provide security and efficiency analysis for DS-CLSPE. The analysis results show that DS-CLSPE can resist chosen keyword attacks (CKA) and has better efficiency than other related schemes.

1. Introduction

The gradual maturity of communication technology, especially the emergence of the 5-th generation wireless systems, has greatly promoted the popularization of IoT, which connects everything to the Internet for intelligent identification, tracking, monitoring, etc. [1–3]. Industrial IoT (IIoT) is one of the main application directions of IoT. It can collect the industrial data in real-time via various sensing devices (e.g., global positioning system and radio frequency identification), realize the optimal utilization of resources, improve the quality, and reduce the cost of the product through further analyzing those collected data [4].

IIoT provides new technological guidance for the development of industry and has great application potential. Nowadays, the IIoT market is expanding rapidly [5]. Meanwhile, the number of IIoT devices will also grow several times, inevitably leading to an explosion of the collected industrial data. In order to effectively manage and utilize these big data, users are more willing to outsource the data to the cloud server provider (CSP), which has powerful data storage and analytical capabilities [6]. In a cloud-based IIoT setting, as presented in Figure 1, massive industrial data is collected through sensor devices and transmitted to the CSP in real-time via the Internet. The CSP further analyzes and mines these data to provide better intelligent services for industrial sectors such as intelligent logistics and manufacturing.

Users enjoy various convenient services offered by CSP; however, their data security and privacy are seriously threatened [7–10]. One reason is that their data is transmitted on an open channel, so an adversary can eavesdrop on the transmitted data to obtain information about an enterprise's production or operations [11]. The other reason is that they will not be able to physically control the outsourced data nor will they be able to fully trust the CSP. For example, the CSP may exploit users' outsourced data to make illegal profits or carry

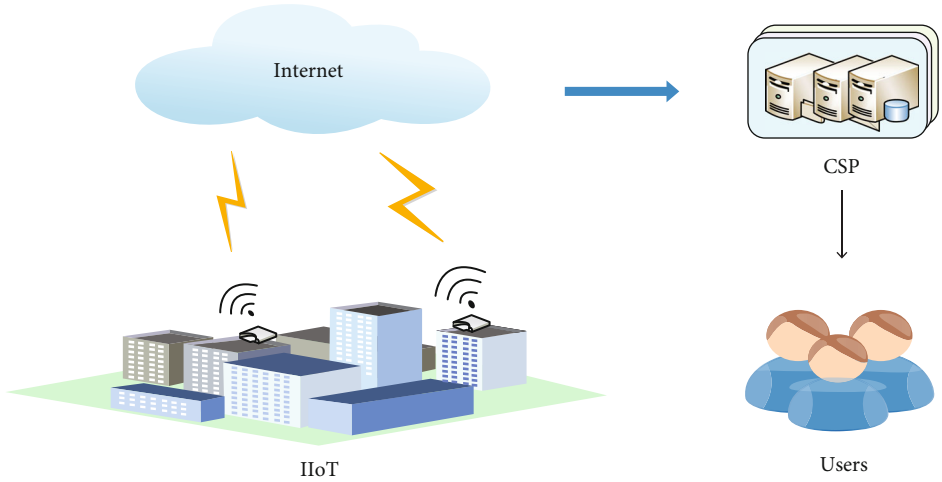

Figure 1: Cloud-based IIoT architecture.

out other malicious acts, such as tampering and deleting. Once the confidential data is eavesdropped or destroyed, it may cause unpredictable damage to an enterprise (e.g., huge economic losses). Overall, it is urgent to establish a practical mechanism with security and privacy preservation for IIoT data utilization and management.

Encryption is the most direct approach to guarantee data privacy. Users can first encrypt the confidential data and then submit the ciphertext to CSP. Although traditional encryption is effective in preserving the privacy of IIoT data, it also incurs some troubles in data utilizations, especially the problem of searching over encrypted data. Since the original data structure will be changed once it is encrypted, the search algorithms for plaintext will not be feasible for the encrypted data. To address this issue, the searchable encryption (SE) technology has emerged, which supports search over ciphertext according to keywords [12, 13]. The first symmetric SE (SSE) scheme was presented in [12]. However, SSE suffers from the troublesome key distribution. To resolve this problem, a SE scheme based on a public-key cryptosystem (SPE) was designed in [13]. Since then, various SPE schemes have been designed [14–17]. However, these SPE schemes face cumbersome certificate management or key escrow burden since the inherent designing structure based on public key infrastructure (PKI) and identity-based cryptosystem. The CLSPE schemes [18–20] can overcome these cumbersome burdens. However, most of the previous CLSPE schemes were computationally expensive since the use of many complex pairing operations. Recently, Lu and Li [21] presented a new CLSPE scheme without pairing operation. Unfortunately, we analyze their scheme suffer from user impersonation attack. To improve the security, we design a new DS-CLSPE scheme without pairing for cloud-based IIoT deployment.

1.1. Related Work. The SE technology provides the functionality of searching over ciphertext without losing the confidentiality of original data. The first concrete SE scheme was presented in [12], which was built on a symmetric cryptosystem. Later, various SSE schemes have been designed. A SSE scheme with verifiable functionality (VSSE) was presented in [22], which can both protect data privacy and provide the verifiability. Recently, Zuo et al. [23] gave two dynamic SSE schemes (i.e., Scheme-A and Scheme-B), which support range queries. The former scheme has forward security property but imposes a heavy storage cost on the client, and the latter scheme reduces the client's storage but loses the forward security. Later, a novel VSSE scheme with forward security was presented in [24]. However, all SSE schemes face the complex key management issue.

To avoid the key management issue, the concept of SPE was introduced in [13]. A SPE system contains three participants: cloud server (CS), data sender (DS), and data receiver (DR). DS uses DR's public key to encrypt his/her own data, including files and keywords extracted from files, and sends the encrypted data to CS. DR uses its own private key to produce a trapdoor for the keyword to be retrieved and submits the trapdoor to CS. Then, CS verifies whether the trapdoor matches the ciphertext and returns the successfully matched files to DR. Later, a SPE scheme with a designated server (dSPE) was constructed, in which only the specified server can run the test algorithm [14]. Lin et al. [25] designed a novel blockchain-based system for the secure outsourcing of bilinear pairings to remove the secure channel and the trusted server. To resist inside keyword guessing attack (IKGA), an authenticated SPE scheme was constructed in [15]. Recently, a SPE scheme with forward security was designed in [16] to resist file-injection attacks [26]. However, the above-proposed schemes are built on PKI cryptosystem; they inevitably face the complicated certificate management problem.

To reduce the overhead of managing certificate, an ID-based cryptosystem was introduced, in which the participant's public key is set to some public information (e.g., name and office number), and the private key is created by a key generation center (KGC) based on the public information [27]. A general framework for transforming a two-level anonymous ID-based encryption (IBE) scheme to an ID-based SPE (IBSPE) scheme was presented in [17]. Recently, Lu et al. [28] constructed an IBSPE with a designated server (dIBSPE), which supports conjunctive keyword search. Li et al. [29] designed two-authenticated dIBSPE schemes based

on symmetric bilinear pairing and asymmetric bilinear pairing, respectively. In their schemes, any adversary cannot run the encryption algorithm to get a valid ciphertext unless it can capture the data sender's private key, and no adversary can perform the test algorithm correctly unless it has the ability to access the specified server's private key. Zhang et al. [30] constructed a proxy-oriented IBSPE scheme based on lattices to resist quantum computer attacks, in which the original data sender delegates his/her own data to a proxy for encryption in order to lower the computation cost of himself. However, all ID-based SPE schemes are plagued by key escrow issues.

In certificateless cryptosystem (CLC), the private key of the participant is jointly created by KGC and the participant itself, which resolves the burden of key escrow and certificate management existing in PKI-based and ID-based cryptosystem [31]. Peng et al. [18] introduced SE technology into the CLC system and proposed a CLSPE scheme with a designated server (dCLSPE). To lower computation overhead in [18], Islam et al. [19] designed a new dCLSPE scheme based on the problems of CDH and BDH. He et al. [20] constructed a novel-authenticated CLSPE scheme against IKGA attacks. However, all previous CLSPE schemes involve bilinear pairing operations, which require high computation overhead. Recently, a CLSPE scheme without pairing operation is designed in [21]. Unfortunately, we analyze scheme [21] cannot resist user impersonation attacks, thus, we develop a new pairing-free CLSPE scheme.

1.2. Research Contributions. An efficient pairing-free DS-CLSPE scheme for cloud-based IIoT is designed in this paper. The main research contributions are listed below:

(i) First, we analyze Lu et al.'s scheme is subject to user impersonation attack, and their scheme requires a secure channel for trapdoor transmission

(ii) Second, we give the system model for DS-CLSPE and construct a new DS-CLSPE scheme, which not only eliminates the need for bilinear pairings but also removes the use of secure channel

(iii) Finally, we present a security analysis for DS-CLSPE and show it can resist the CKA attack. Furthermore, we evaluate the efficiency of DS-CLSPE in terms of computation and communication costs

1.3. Organization of the Paper. The following sections are arranged as below. Section 2 presents some preliminary knowledge. Section 3 shows the analysis of Lu et al.'s scheme. Section 4 gives the detailed construction of our DS-CLSPE scheme. Section 5 and Section 6 show the security proof and the performance analysis of DS-CLSPE, respectively. The last section is mainly to summarize the full paper.

2. Preliminaries

We first present the complexity assumptions used in this paper and then give the system model and formal definition of DS-CLSPE.

2.1. Complexity Assumptions. Suppose $\mathbb{G}$ denotes a q-order cyclic group, and the point $P \in \mathbb{G}$ denotes a generator.

2.1.1. Computational Diffie-Hellman (CDH) Assumption. Given three points P, aP, $bP \in \mathbb{G}$ ($a, b \in \mathbb{Z}_q^*$ are unknown numbers), to figure out abP.

2.2. System Model. As presented in Figure 2, a DS-CLSPE system contains five participants: data sender (DS), data receiver (DR), KGC, front server (FS), and back server (BS). The responsibilities of each participant are described below.

(i) KGC is responsible for generating system parameters and participants' partial keys

(ii) DS encrypts his/her own data and then outsources the encrypted data to FS and BS

(iii) DR submits a trapdoor to FS for querying the encrypted data

(iv) FS generates an intermediate testing-state ciphertext C_{ITS} according to the received trapdoor and submits C_{ITS} to BS

(v) BS generates the final test results according to C_{ITS} and returns the test results to DR

2.3. Formal Definition. A DS-CLSPE scheme contains the following algorithms.

(i) Setup(λ): this algorithm is implemented by KGC. Inputs a security parameter λ, returns the system master key s and public parameters parm

(ii) PartialKeyGen(parm,s,ID_i): this algorithm is performed by KGC. Inputs parm, s, and the identity $\mathrm{ID}_i (i \in \{\mathrm{DR}, \mathrm{FS}, \mathrm{BS}\})$, outputs the partial key pair (T_i, d_i) for the corresponding participant

(iii) KeyGen(parm,ID_i,T_i,d_i): each participant generates its own full public/private keys $\mathrm{PK}_i/\mathrm{SK}_i$ by performing this algorithm

(iv) Encrypt(parm,w,ID_i,PK_i): DS executes this algorithm to generate the ciphertext C_w for keyword w

(v) Trapdoor(parm,w',$\mathrm{SK}_{\mathrm{DR}}$,$\mathrm{PK}_{\mathrm{FS}}$,$\mathrm{PK}_{\mathrm{BS}}$): DR performs this algorithm to obtain the trapdoor $T_w{}'$ for w'.

(vi) FrontTest(parm,C_w,$T_w{}'$,$\mathrm{SK}_{\mathrm{FS}}$): FS performs this algorithm to generate an intermediate testing-state C_{ITS}

(vii) BackTest(parm,C_{ITS},$\mathrm{SK}_{\mathrm{BS}}$): this algorithm is performed by BS. Inputs parm, C_{ITS}, $\mathrm{SK}_{\mathrm{BS}}$, outputs "1" if the test succeeds and "0" otherwise

3. Weakness of Lu et al.'s Scheme

We first present Lu et al.'s scheme and then analyze its security weakness.

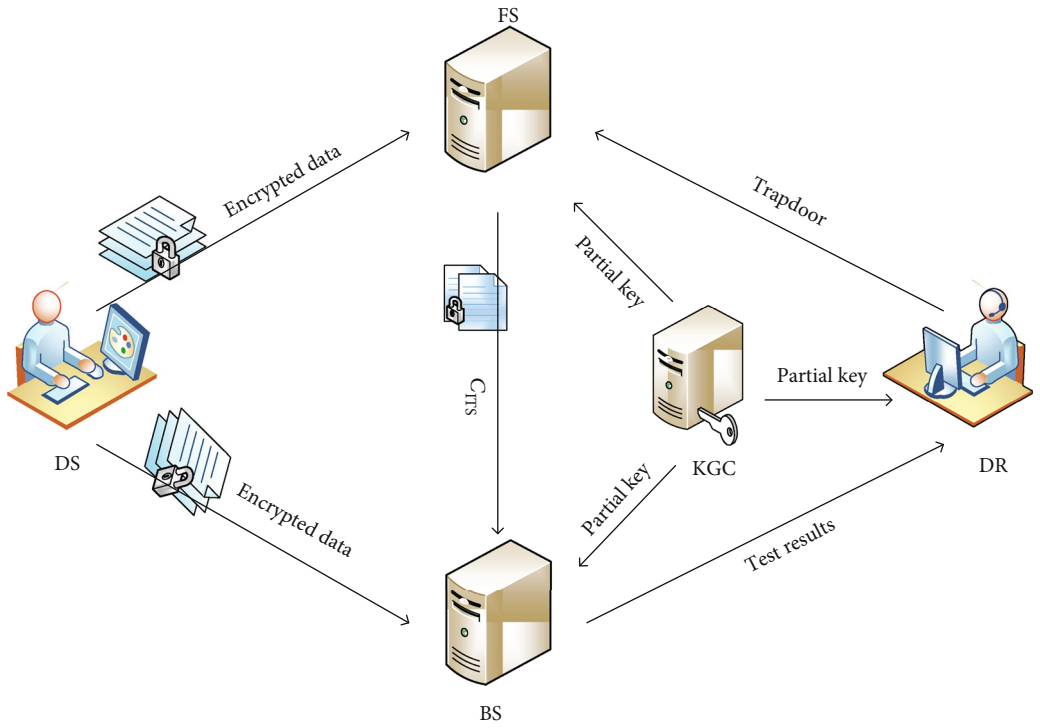

Figure 2: The system model for DS-CLSPE.

3.1. Review of Lu et al.'s Scheme. The detailed construction of Lu et al.'s scheme is as follows.

(i) Setup(λ): suppose q is a large prime number and $\mathbb{G}$ is a group with order q. P denotes a generator selected from $\mathbb{G}$, and $h_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$, $h_2 : \{0,1\}^* \longrightarrow \mathbb{Z}_q^*$, and $h_3 : \mathbb{G} \longrightarrow \mathbb{Z}_q^*$ denote three different hash functions. KGC performs this algorithm as follows

(1) Selects $s \in \mathbb{Z}_q^*$ at random

(2) Calculates $P_{\text{pub}} = sP$

(3) Publishes $\text{parm} = \{\mathbb{G}, P, P_{\text{pub}}, q, h_1, h_2, h_3\}$ and keeps s in secret

(ii) PartialKeyGen(parm, s, ID_{DR}): KGC selects $t_{\text{DR}} \in \mathbb{Z}_q^*$, computes $T_{\text{DR}} = t_{\text{DR}}P$, $d_{\text{DR}} = t_{\text{DR}} + sh_1(\text{ID}_{\text{DR}})$, and sends the partial key pair $\{T_{\text{DR}}, d_{\text{DR}}\}$ to the receiver

(iii) KeyGen(parm, ID_{DR}, T_{DR}, d_{DR}): the receiver picks $x_{\text{DR}} \in \mathbb{Z}_q^*$, calculates $P_{\text{DR}} = x_{\text{DR}}P$, and sets $\text{PK}_{\text{DR}} = (T_{\text{DR}}, P_{\text{DR}})$, $\text{SK}_{\text{DR}} = (d_{\text{DR}}, x_{DR})$ as his/her own public/private keys

(iv) Encrypt(parm, w, PK_{DR}): the sender randomly chooses $r \in \mathbb{Z}_q^*$, calculates $C_w^1 = rP$ and

$$C_w^2 = h_3(rh_2(w)Q_{\text{DR}}), \tag{1}$$

here $Q_{\text{DR}} = T_{\text{DR}} + P_{\text{DR}} + h_1(\text{ID}_{\text{DR}})P_{\text{pub}}$, and sets the ciphertext $C_w = (C_w^1, C_w^2)$.

(v) Trapdoor(parm, w', SK_{DR}): the receiver computes

$$T_w{}' = (d_{DR} + x_{DR})h_2\left(w'\right). \tag{2}$$

(vi) Test(parm, C_w, $T_w{}'$): given C_w and $T_w{}'$, the server checks

$$C_w^2 \stackrel{?}{=} h_3\left(T_w{}'C_w^1\right). \tag{3}$$

If this equation holds, returns "1"; Otherwise, returns "0".

3.2. Attack on Lu et al.'s Scheme. Lu et al.'s scheme cannot prevent user impersonation attack launched by TypeI adversary $\mathcal{A}_1$. During the attack, $\mathcal{A}_1$ can first forge the private key of the receiver ID_{DR} and then impersonate ID_{DR} to calculate a trapdoor T_w for a challenge keyword w using the forged private key. The detail attack is presented below.

(i) *Setup*: the challenger $\mathcal{C}$ generates $\{\text{parm}, s\}$ by performing the Setup algorithm

(ii) *Queries*: $\mathcal{A}_1$ with identity $\text{ID}_{\mathcal{A}_1}$ can ask the following queries

(1) *Extract Partial Private Key Query*: if $\mathcal{A}_1$ submits this query, then $\mathcal{C}$ returns $\{d_{\mathcal{A}_1}, T_{\mathcal{A}_1}\}$ to $\mathcal{A}_1$, where $d_{\mathcal{A}_1} = t_{\mathcal{A}_1} + sh_1(\text{ID}_{\mathcal{A}_1})$ and $T_{\mathcal{A}_1} = t_{\mathcal{A}_1}P$

Upon receiving $\{d_{\mathscr{A}_1}, T_{\mathscr{A}_1}\}$, $\mathscr{A}_1$ can forge the partial key pair $(d_{\mathrm{DR}}, T_{\mathrm{DR}})$ of the receiver $\mathrm{ID}_{\mathrm{DR}}$ as follows:

$$T^*_{\mathrm{DR}} = T_{\mathscr{A}_1} \cdot h_1(\mathrm{ID}_{\mathrm{DR}}) \cdot h_1\left(\mathrm{ID}_{\mathscr{A}_1}\right)^{-1}, \quad (4)$$

$$d^*_{DR} = d_{\mathscr{A}_1} \cdot h_1(ID_{DR}) \cdot h_1\left(ID_{\mathscr{A}_1}\right)^{-1}. \quad (5)$$

(2) *Replace Public Key Query*: $\mathscr{A}_1$ selects $x^*_{DR} \in \mathbb{Z}^*_q$ randomly and computes $P^*_{\mathrm{DR}} = x^*_{\mathrm{DR}}P$. Then, $\mathscr{A}_1$ submits this query with $(\mathrm{ID}_{\mathrm{DR}}, \mathrm{PK}^*_{\mathrm{DR}})$, here $\mathrm{PK}^*_{\mathrm{DR}} = (T^*_{\mathrm{DR}}, P^*_{\mathrm{DR}})$. $\mathscr{C}$ will set $\mathrm{PK}_{\mathrm{DR}} \longleftarrow \mathrm{PK}^*_{\mathrm{DR}}$

(iii) *Forge Trapdoor*: once $\mathscr{A}_1$ has successfully forged d^*_{DR} and replaced $\mathrm{PK}_{\mathrm{DR}}$, it can forge a trapdoor T_{w_0} for a keyword w_0 with respect to the identity $\mathrm{ID}_{\mathrm{DR}}$ as below:

$$T_{w_0} = (d^*_{\mathrm{DR}} + x^*_{\mathrm{DR}})h_2(w_0). \quad (6)$$

(iv) *Challenge*: $\mathscr{A}_1$ returns the keywords (w_0, w_1) and the identity $\mathrm{ID}_{\mathrm{DR}}$ as challenge target. Then $\mathscr{C}$ picks up $\mu \in \{0, 1\}$ randomly and returns $C_{w_\mu} \longleftarrow \mathrm{Encrypt}(\mathrm{parm}, w_\mu, \mathrm{PK}_{\mathrm{DR}})$ to $\mathscr{A}_1$

Upon receiving C_{w_μ}, $\mathscr{A}_1$ performs Test algorithm and gets $\tau \longleftarrow \mathrm{Test}(\mathrm{parm}, C_{w_\mu}, T_{w_0}) \in \{0, 1\}$.

(v) *Guess*: $\mathscr{A}_1$ returns $\mu' = 0$ if $\tau = 1$; otherwise, $\mathscr{A}_1$ returns $\mu' = 1$. It is easy to see that $\mu' = \mu$, i.e., the advantage of $\mathscr{A}_1$ is always 1

4. The Proposed DS-CLSPE Scheme

To overcome the weakness of Lu et al.'s scheme and avoid the use of bilinear pairing, we develop a new dual-server CLSPE scheme. The details are described as follows.

(i) Setup(λ): suppose q is a large prime number, $\mathbb{G}$ is a group with order q. Let P denote a generator of $\mathbb{G}$, and $h_0 : \{0,1\}^* \times \mathbb{G} \longrightarrow \mathbb{Z}^*_q$, $h_1 : \{0,1\}^* \times \mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{Z}^*_q$, and $h_2 : \{0,1\}^* \times \{0,1\}^* \longrightarrow \mathbb{Z}^*_q$ denote three different hash functions. KGC performs the following steps

(1) Chooses $s \in \mathbb{Z}^*_q$ randomly

(2) Calculates $P_{\mathrm{pub}} = sP$

(3) Publishes $\mathrm{parm} = \{P, P_{\mathrm{pub}}, \mathbb{G}, h_0, h_1, h_2\}$ and keeps s secretly

(ii) PartialKeyGen(parm, s, ID_i): takes parm, s, and the identity $\mathrm{ID}_i (i \in \{\mathrm{DR}, \mathrm{FS}, \mathrm{BS}\})$ as inputs, KGC performs this algorithm as follows

(1) Selects $t_i \in \mathbb{Z}^*_q$

(2) Calculates $T_i = t_iP$, $d_i = t_i + s\alpha_i \bmod q$, where $\alpha_i = h_0(\mathrm{ID}_i, T_i)$.

(3) Sends $\{T_i, d_i\}$ to the corresponding participant

(iii) KeyGen(parm, ID_i, T_i, d_i): this algorithm is performed by the participant $i \in \{\mathrm{DR}, \mathrm{FS}, \mathrm{BS}\}$ as follows

(1) Chooses $x_i \in \mathbb{Z}^*_q$ randomly

(2) Calculates $P_i = x_iP$

(3) Sets $\mathrm{PK}_i = (T_i, P_i)$ and $\mathrm{SK}_i = (d_i, x_i)$

(iv) Encrypt(parm, w, ID_i, PK_i): given ID_i, $\mathrm{PK}_i = (T_i, P_i)$ $(i \in \{\mathrm{DR}, \mathrm{FS}, \mathrm{BS}\})$ and keyword w. DR performs the steps below to produce ciphertext C_w

(1) Selects $r \in \mathbb{Z}^*_q$ randomly

(2) Calculates $C^1_w = rP$

(3) Calculates

$$C^2_w = rQ_{\mathrm{FS}} + rQ_{\mathrm{BS}} + h_2(w, \mathrm{ID}_{\mathrm{DR}})Q_{\mathrm{DR}}, \quad (7)$$

where

$$Q_i = T_i + \beta_iP_i + \alpha_iP_{\mathrm{pub}}, \quad (8)$$

$$\alpha_i = h_0(\mathrm{ID}_i, T_i), \quad (9)$$

$$\beta_i = h_1(\mathrm{ID}_i, T_i, P_i). \quad (10)$$

(4) Sets $C_w = (C^1_w, C^2_w)$

The parameters $\{Q_i, \alpha_i, \beta_i\}$ can be published publicly.

(v) Trapdoor(parm, w', $\mathrm{SK}_{\mathrm{DR}}$, $\mathrm{PK}_{\mathrm{FS}}$, $\mathrm{PK}_{\mathrm{BS}}$): the receiver generates the trapdoor $T_{w'}$ of keyword w' as below

(1) Selects $l \in \mathbb{Z}^*_q$ randomly

(2) Computes $T^1_{w'} = lP$

(3) Computes

$$T^2_{w'} = (d_{\mathrm{DR}} + \beta_{\mathrm{DR}}x_{\mathrm{DR}})h_2\left(w', \mathrm{ID}_{\mathrm{DR}}\right)Q_{\mathrm{FS}} + lQ_{\mathrm{BS}}, \quad (11)$$

where Q_i and β_i are computed as above.

(4) Sends $T_{w'} = (T^1_{w'}, T^2_{w'})$ to the front server

(vi) FrontTest(parm, C_w, $T_{w'}$, $\mathrm{SK}_{\mathrm{FS}}$): given the ciphertext C_w and a trapdoor $T_{w'}$, the front server FS performs the following steps to generate intermediate testing-state C_{ITS}

(1) Selects $\gamma \in \mathbb{Z}^*_q$ randomly

(2) Calculates $C^* = (d_{\mathrm{FS}} + \beta_{\mathrm{FS}}x_{\mathrm{FS}})C^1_w$

(3) Calculates

$$C_1^* = \gamma\left(C^* - T_{w'}^1\right), \tag{12}$$

$$C_2^* = \gamma\left((d_{FS} + \beta_{FS}x_{FS})\left(C_w^2 - C^*\right) - T_{w'}^2\right). \tag{13}$$

(4) Sends $C_{ITS} = (C_1^*, C_2^*)$ to the back server BS

(vii) BackTest(parm, C_{ITS}, SK_{BS}): the back server checks

$$C_2^* \stackrel{?}{=} (d_{BS} + \beta_{BS}x_{BS})C_1^*. \tag{14}$$

If this equation holds, BS returns "1"; otherwise, returns "0".

Correctness. Suppose that $w = w'$, then we have

$$C^* = (d_{FS} + \beta_{FS}x_{FS})C_w^1 = rQ_{FS}, \tag{15}$$

$$C_1^* = \gamma\left(C^* - T_{w'}^1\right) = \gamma(rQ_{FS} - lP), \tag{16}$$

$$\begin{aligned} C_2^* &= \gamma\left((d_{FS} + \beta_{FS}x_{FS})\left(C_w^2 - C^*\right) - T_{w'}^2\right) \\ &= \gamma\left((d_{FS} + \beta_{FS}x_{FS})(rQ_{BS} + h_2(w, ID_{DR})Q_{DR}) - T_{w'}^2\right) \\ &= \gamma(r(d_{FS} + \beta_{FS}x_{FS})Q_{BS} - lQ_{BS}) \\ &= \gamma(r(d_{BS} + \beta_{BS}x_{BS})Q_{FS} - lQ_{BS}) \\ &= \gamma(d_{BS} + \beta_{BS}x_{BS})(rQ_{FS} - lP) \\ &= (d_{BS} + \beta_{BS}x_{BS})C_1^*. \end{aligned} \tag{17}$$

5. Security Analysis

This section first presents the security model of DS-CLSPE and then gives the formal proof of DS-CLSPE.

5.1. Security Model. Two types of adversaries should be considered in a certificateless cryptosystem [31, 32].

Type 1. This adversary is denoted as $\mathcal{A}_1$, who has no master key but can replace anyone's public key.

Type II. This adversary is denoted as $\mathcal{A}_2$, who holds the master key but cannot replace anyone's public key.

As defined in scheme [33], assume that both BS and FS are honest-but-curious, and that they cannot collude. The security model of DS-CLSPE is defined by the following game, i.e., indistinguishability against CKA attack (IND-CKA), which is the interaction between an adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ and a challenger $\mathcal{C}$.

Definition 1 (IND-CKA). The DS-CLSPE scheme is said IND-CKA secure if $\mathcal{A}$'s advantage of winning in the following game is negligible.

Game. This game is interacted between $\mathcal{A}$ and $\mathcal{C}$.

(i) *Setup*: $\mathcal{C}$ generates $\{parm, s\}$ by executing Setup algorithm. If $\mathcal{A} = \mathcal{A}_1$, $\mathcal{C}$ returns parm to $\mathcal{A}$; otherwise ($\mathcal{A} = \mathcal{A}_2$), returns both parm and s

(ii) *Phase* 1: the following oracles can be queried by $\mathcal{A}$

(a) *CreateUser Queries*: upon receiving this query for ID_i, $\mathcal{C}$ checks whether ID_i has been created. If so, $\mathcal{C}$ outputs PK_i directly. Otherwise, $\mathcal{C}$ performs the algorithm KeyGen to produce the key pair (PK_i, SK_i) and returns PK_i

(b) *PrivateKey Queries*: upon receiving this query for ID_i, $\mathcal{C}$ checks whether ID_i has been created. If so, $\mathcal{C}$ returns SK_i; otherwise, returns $\perp$

(c) *PatialPrivateKey Queries*: upon receiving this query for ID_i, $\mathcal{C}$ checks whether ID_i has been created. If so, $\mathcal{C}$ returns partial private key d_i; otherwise, returns $\perp$

(d) *ReplacePublicKey Queries*: if $\mathcal{A} = \mathcal{A}_1$, then it can perform these queries. Upon receiving $\mathcal{A}$'s query for ID_i with a false public key PK_i^*, $\mathcal{C}$ sets $PK_i \longleftarrow PK_i^*$

(e) *Trapdoor Queries*: if $\mathcal{A}$ submits this query for (ID_i, w), then $\mathcal{C}$ returns the corresponding trapdoor T_w if ID_i has been created; otherwise, returns $\perp$

(iii) *Challenge*: $\mathcal{A}$ selects identity ID_{ch} and keywords (w_0, w_1) as challenge targets. $\mathcal{C}$ picks up $\sigma \in \{0, 1\}$ at random and returns $C_{w_\sigma} \longleftarrow$ Encrypt(parm, w_σ, ID_{ch}, PK_{ch}, PK_{FS}, PK_{BS}) to $\mathcal{A}$

(iv) *Phase* 2: the oracles defined as in phase 1 can be continuously queried by $\mathcal{A}$

(v) *Guess*: $\mathcal{A}$ returns $\sigma' \in \{0, 1\}$. We say $\mathcal{A}$ wins the above game if $\sigma' = \sigma$ and below conditions hold: (1) *Trapdoor* queries for w_0 and w_1 have never been submitted by $\mathcal{A}$; (2) If $\mathcal{A} = \mathcal{A}_1$, it has never made *PrivateKey* and *PartialPrivateKey* queries for ID_{ch}; if $\mathcal{A} = \mathcal{A}_2$, it has never made *PrivateKey* queries for ID_{ch} (note that $\mathcal{A}_2$ can calculate partial private key as it knows master key).

Let $Adv_{\mathcal{A}}(\lambda) = |Pr[\sigma' = \sigma] - 1/2|$ express $\mathcal{A}$'s advantage in the above-defined game.

5.2. Provable Security

Theorem 2. *The DS-CLSPE scheme can achieve IND-CKA secure if the CDH problem is difficult to solve.*

Theorem 2 can be proofed by the two lemmas below.

Lemma 3. *Let the advantage of $\mathcal{A}_1$ winning the IND-CKA game be ε. Then, we can construct an algorithm $\mathcal{C}$ to calculate the CDH problem with advantage*

$$\varepsilon' \geq \frac{\varepsilon}{(q_T + q_{SK} + q_{PSK} + 1)e}, \tag{18}$$

where e is Euler number and q_{SK}, q_{PSK}, q_T denote the maximum number of PrivateKey queries, PartialPrivateKey queries, and Trapdoor queries, respectively.

Proof. Let q be a large prime number. Given (P, aP, bP), $\mathscr{C}$'s goal is to output abP.

(i) *Setup*: $\mathscr{C}$ generates the public parameters $\text{parm} = \{\mathbb{G}, P, P_{\text{pub}}, h_0, h_1, h_2, q\}$, here $P_{\text{pub}} = aP$. $\mathscr{C}$ selects $x_{\text{FS}}, x_{\text{BS}}, t_{\text{FS}}, t_{\text{BS}} \in \mathbb{Z}_q^*$, computes $P_{\text{FS}} = x_{\text{FS}}P$, $P_{\text{BS}} = x_{\text{BS}}P$, $T_{\text{FS}} = t_{\text{FS}}P$, and $T_{\text{BS}} = t_{\text{BS}}P$, and sets $\text{PK}_{\text{FS}} = (T_{\text{FS}}, P_{\text{FS}})$, $\text{PK}_{\text{BS}} = (T_{\text{BS}}, P_{\text{BS}})$. Finally, $\mathscr{C}$ sends $\{\text{parm}, \text{PK}_{\text{FS}}, \text{PK}_{\text{BS}}\}$ to $\mathscr{A}_1$

(ii) *Phase* 1: the following oracles can be queried by $\mathscr{A}_1$

(a) h_0 *Queries*: a h_0-list with tuples $(\text{ID}_i, T_i, \alpha_i)$ is maintained by $\mathscr{C}$. When $\mathscr{A}_1$ performs h_0 query for (ID_i, T_i), $\mathscr{C}$ first checks whether $(\text{ID}_i, T_i, \alpha_i)$ is already in h_0-list. If so, $\mathscr{C}$ returns α_i; otherwise, $\mathscr{C}$ selects $\alpha_i \in \mathbb{Z}_q^*$ at random, adds $(\text{ID}_i, T_i, \alpha_i)$ into h_0-list, and returns α_i

(b) h_1 *Queries*: $\mathscr{C}$ maintains a h_1-list with tuples $(\text{ID}_i, T_i, P_i, \beta_i)$. Upon receiving $\mathscr{A}_1$'s query for (ID_i, T_i, P_i), $\mathscr{C}$ searches $(\text{ID}_i, T_i, P_i, \beta_i)$ from h_1-list. If h_1-list already contains the searched tuple, $\mathscr{C}$ outputs β_i directly; otherwise, $\mathscr{C}$ chooses $\beta_i \in \mathbb{Z}_q^*$ at random, adds $(\text{ID}_i, T_i, P_i, \beta_i)$ into h_1-list, and returns β_i

(c) h_2 *Queries*: $\mathscr{C}$ maintains a h_2-list with tuples $(w_i, \text{ID}_i, h_{2i})$. Upon receiving this query for (w_i, ID_i), $\mathscr{C}$ searches $(w_i, \text{ID}_i, h_{2i})$ from h_2-list and returns h_{2i} directly if $(w_i, \text{ID}_i, h_{2i})$ already exists in h_2-list; otherwise, $\mathscr{C}$ selects $h_{2i} \in \mathbb{Z}_q^*$ at random, adds $(w_i, \text{ID}_i, h_{2i})$ into h_2-list, and returns h_{2i}

(d) *CreateUser Queries*: $\mathscr{C}$ maintains a user-list with tuples $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$. Upon receiving $\mathscr{A}_1$'s query for ID_i, $\mathscr{C}$ searches user-list for $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$. $\mathscr{C}$ outputs PK_i directly if user-list already includes the searched tuple; otherwise, $\mathscr{C}$ tosses a $\text{coin}_i \in \{0, 1\}$ at random such that $\Pr[\text{coin}_i = 1] = \delta$ (δ will be computed later) and executes the following steps

(1) If $\text{coin}_i = 1$, $\mathscr{C}$ selects two random numbers t_i, x_i from $\mathbb{Z}_q^*$, sets $\text{PK}_i = (T_i, P_i)$, where $T_i = t_iP$, $P_i = x_iP$. $\mathscr{C}$ adds $(\text{ID}_i, \text{PK}_i, \perp, t_i, x_i, 1)$ into user-list and returns PK_i

(2) Otherwise ($\text{coin}_i = 0$), $\mathscr{C}$ selects three numbers x_i, d_i, α_i from $\mathbb{Z}_q^*$, sets $\text{SK}_i = (d_i, x_i)$ and $\text{PK}_i = (T_i, P_i)$, where $T_i = d_iP - \alpha_iP_{\text{pub}}$, $P_i = x_iP$. $\mathscr{C}$ adds $(\text{ID}_i, \text{PK}_i, \text{SK}_i, \perp, \perp, 0)$ into user-list, adds $(\text{ID}_i, T_i, \alpha_i)$ into h_0-list, and returns PK_i

(e) *PrivateKey Queries*: when $\mathscr{A}_1$ submits this query for ID_i, $\mathscr{C}$ searches $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$ from user-list. If $\text{coin}_i = 0$, $\mathscr{C}$ outputs SK_i; otherwise, $\mathscr{C}$ aborts the game (this event is denoted as Event_1).

(f) *PartialPrivateKey Queries*: when $\mathscr{A}_1$ submits this query for ID_i, $\mathscr{C}$ searches $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$ from user-list. If $\text{coin}_i = 0$, $\mathscr{C}$ outputs the first part of SK_i, i.e., d_i; otherwise, $\mathscr{C}$ aborts the game (this event is denoted as Event_2).

(g) *ReplacePublicKey Queries*: when $\mathscr{A}_1$ submits this query with a value PK_i^*, $\mathscr{C}$ replaces PK_i with PK_i^*. Note that this query implies $\mathscr{A}_1$ must also submit the corresponding private key SK_i^*

(h) *Trapdoor Queries*: when $\mathscr{A}_1$ submits this query for (w, ID_i), $\mathscr{C}$ searches $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$ from user-list and recovers $(\text{ID}_i, T_i, P_i, \beta_i)$ and $(w_i, \text{ID}_i, h_{2i})$ from h_1-list and h_2-list, respectively. If $\text{coin}_i = 0$, $\mathscr{C}$ chooses $l \in \mathbb{Z}_q^*$ and computes $T_w^1 = lP$, $T_w^2 = (d_i + \beta_i x_i)h_{2i}Q_{\text{FS}} + lQ_{\text{BS}}$, where Q_{FS} and Q_{BS} are computed as in the proposed scheme. $\mathscr{C}$ returns $T_w = (T_w^1, T_w^2)$. Otherwise, $\mathscr{C}$ aborts the game (this event is denoted as Event_3).

(iii) *Challenge*: $\mathscr{A}_1$ outputs ID_{ch} and (w_0, w_1). $\mathscr{C}$ recovers $(\text{ID}_{\text{ch}}, \text{PK}_{\text{ch}}, \text{SK}_{\text{ch}}, t_{\text{ch}}, x_{\text{ch}}, \text{coin}_{\text{ch}})$ and $(\text{ID}_{\text{ch}}, T_{\text{ch}}, P_{\text{ch}}, \beta_{\text{ch}})$ from user-list and h_1-list, respectively. If $\text{coin}_{\text{ch}} = 1$, $\mathscr{C}$ randomly selects $\sigma \in \{0, 1\}$, $C_{w_\sigma}^2 \in \mathbb{G}$, recovers $(w_\sigma, \text{ID}_{\text{ch}}, h_{2\sigma})$ from h_2-list, sets $C_{w_\sigma}^1 = bP$, and returns $C_{w_\sigma} = (C_{w_\sigma}^1, C_{w_\sigma}^2)$ to $\mathscr{A}_1$. Otherwise, $\mathscr{C}$ ends the game (this event is denoted as Event_4).

Note that $C_{w_\sigma}^2$ is implicitly defined as $bQ_{\text{FS}} + bQ_{\text{BS}} + h_{2\sigma}Q_{\text{ch}} = b(T_{\text{FS}} + \beta_{\text{FS}}P_{\text{FS}} + \alpha_{\text{FS}}P_{\text{pub}}) + b(T_{\text{BS}} + \beta_{\text{BS}}P_{\text{BS}} + \alpha_{\text{BS}}P_{\text{pub}}) + h_{2\sigma}(T_{\text{ch}} + \beta_{\text{ch}}P_{\text{ch}} + \alpha_{\text{ch}}P_{\text{pub}})$.

(iv) *Phase* 2: The oracles defined in phase1 can be asked continuously by $\mathscr{A}_1$

(v) *Guess*: $\mathscr{A}_1$ outputs σ'. If $\sigma' = \sigma$, then $\mathscr{C}$ wins in the above game

At this point, $\mathscr{C}$ can compute the value abP as follows: $Z = (\alpha_{\text{FS}} + \alpha_{\text{BS}})^{-1}(C_{w_\sigma}^2 - h_{2\sigma}(T_{\text{ch}} + \beta_{\text{ch}}P_{\text{ch}} + \alpha_{\text{ch}}P_{\text{pub}}) - (t_{\text{FS}} + t_{\text{BS}} + \beta_{\text{FS}}x_{\text{FS}} + \beta_{\text{BS}}x_{\text{BS}})bP) = abP$.

Analysis. The advantage of $\mathscr{C}$ winning the game is analyzed below. $\mathscr{C}$ wins in above game if none of the events Event_i $(i = 1, 2, 3, 4)$ occur.

From the above proof of Lemma 3,

$$\Pr\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3} \wedge \overline{\text{Event}_4}\right] = \delta(1-\delta)^{q_{\text{SK}}+q_{\text{PSK}}+q_T}, \tag{19}$$

when $\delta = 1/(q_{\text{SK}} + q_{\text{PSK}} + q_T + 1)$, this value is at its maximum value

$$\begin{aligned}\Pr&\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3} \wedge \overline{\text{Event}_4}\right] \\ &= \left(\frac{1}{q_{\text{SK}} + q_{\text{PSK}} + q_T + 1}\right)\left(1 - \frac{1}{q_{\text{SK}} + q_{\text{PSK}} + q_T + 1}\right)^{q_{\text{SK}}+q_{\text{PSK}}+q_T} \\ &\approx \frac{1}{(q_T + q_{\text{SK}} + q_{\text{PSK}} + 1)e}.\end{aligned} \tag{20}$$

Thus,

$$\begin{aligned}\varepsilon' &\geq \varepsilon \cdot \Pr\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3} \wedge \overline{\text{Event}_4}\right] \\ &= \frac{\varepsilon}{(q_T + q_{\text{SK}} + q_{\text{PSK}} + 1)e}.\end{aligned} \tag{21}$$

Lemma 4. *Let ε denote the advantage of $\mathcal{A}_2$ winning in IND-CKA game. Then, the algorithm $\mathcal{C}$ can be constructed to solve the CDH problem with advantage*

$$\varepsilon' \geq \frac{\varepsilon}{(q_{SK} + q_T + 1)e}, \tag{22}$$

where e, q_{SK}, and q_T are defined as Lemma 3.

Proof. Given a CDH instance, i.e., (P, aP, bP), $\mathcal{C}$ will try to output abP.

(i) *Setup*: $\mathcal{C}$ selects $s \in \mathbb{Z}_q^*$, calculates $P_{\text{pub}} = sP$ and generates parm $= \{q, \mathbb{G}, P, P_{\text{pub}}, h_0, h_1, h_2\}$. Then, $\mathcal{C}$ selects x_{FS}, x_{BS}, t_{FS}, and $t_{\text{BS}} \in \mathbb{Z}_q^*$, sets $P_{\text{FS}} = x_{\text{FS}}P$, $P_{\text{BS}} = x_{\text{BS}}P$, $T_{\text{FS}} = t_{\text{FS}}aP$, and $T_{\text{BS}} = t_{\text{BS}}aP$, and lets $\text{PK}_{\text{FS}} = (T_{\text{FS}}, P_{\text{FS}})$ and $\text{PK}_{\text{BS}} = (T_{\text{BS}}, P_{\text{BS}})$. Finally, $\mathcal{C}$ sends $\{\text{parm}, s, \text{PK}_{\text{FS}}, \text{PK}_{\text{BS}}\}$ to $\mathcal{A}_2$

(ii) *Phase*1: the following oracles can be queried by $\mathcal{A}_2$

(a) h_0, h_1, h_2 *Queries*: when $\mathcal{A}_2$ submits these oracle queries, $\mathcal{C}$ responds as defined in Lemma 3

(b) *CreateUser Queries*: $\mathcal{C}$ maintains a user-list with $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$. Upon receiving this query for ID_i, $\mathcal{C}$ searches $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$ from user-list. If this tuple is already in user-list, $\mathcal{C}$ returns PK_i; otherwise, $\mathcal{C}$ tosses $\text{coin}_i \in \{0, 1\}$ randomly such that $\Pr[\text{coin}_i = 1] = \delta$ (δ will be computed later) and executes the following steps

(1) If $\text{coin}_i = 1$, $\mathcal{C}$ selects t_i, x_i from $\mathbb{Z}_q^*$, computes $T_i = t_i aP$, $P_i = x_i P$, and sets $\text{PK}_i = (T_i, P_i)$. $\mathcal{C}$ adds $(\text{ID}_i, \text{PK}_i, \perp, \perp, x_i, 1)$ into user-list and returns PK_i

(2) Otherwise, $\mathcal{C}$ selects x_i, t_i, α_i from $\mathbb{Z}_q^*$ at random, computes $d_i = t_i + s\alpha_i \bmod q$, $T_i = t_i P$, $P_i = x_i P$, and sets $\text{SK}_i = (d_i, x_i)$, $\text{PK}_i = (T_i, P_i)$. $\mathcal{C}$ adds $(\text{ID}_i, \text{PK}_i, \text{SK}_i, \perp, \perp, 0)$ into user-list, adds $(\text{ID}_i, T_i, \alpha_i)$ into h_0-list, and returns PK_i

(c) *PrivateKey Queries*: Upon receiving this query for ID_i, $\mathcal{C}$ searches user-list to find $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$. If $\text{coin}_i = 0$, $\mathcal{C}$ outputs SK_i; otherwise, $\mathcal{C}$ aborts the game (this event is denoted as Event_1)

(d) *Trapdoor Queries*: when $\mathcal{A}_2$ submits the query for (w, ID_i), $\mathcal{C}$ searches $(\text{ID}_i, \text{PK}_i, \text{SK}_i, t_i, x_i, \text{coin}_i)$ from user-list and recovers $(\text{ID}_i, T_i, P_i, \beta_i)$ and $(w_i, \text{ID}_i, h_{2i})$ from h_1-list and h_2-list, respectively. If $\text{coin}_i = 0$, $\mathcal{C}$ chooses $l \in \mathbb{Z}_q^*$, computes $T_w^1 = lP$, $T_w^2 = (d_i + \beta_i x_i)h_{2i}Q_{\text{FS}} + lQ_{\text{BS}}$, and returns $T_w = (T_w^1, T_w^2)$ to $\mathcal{A}_2$, where Q_{FS} and Q_{BS} are computed as in the proposed scheme; otherwise, $\mathcal{C}$ aborts the game (this event is denoted as Event_2)

(iii) *Challenge*: $\mathcal{A}_2$ outputs ID_{ch} and (w_0, w_1). $\mathcal{C}$ recovers $(\text{ID}_{\text{ch}}, \text{PK}_{\text{ch}}, \text{SK}_{\text{ch}}, t_{\text{ch}}, x_{\text{ch}}, \text{coin}_{\text{ch}})$ from user-list. If $\text{coin}_{\text{ch}} = 1$, $\mathcal{C}$ randomly selects $\sigma \in \{0, 1\}$, $C_{w_\sigma}^2 \in \mathbb{G}$ and recovers $(w_\sigma, ID_{ch}, h_{2\sigma})$ from h_2-list. $\mathcal{C}$ sets $C_{w_\sigma}^1 = bP$ and returns $C_{w_\sigma} = (C_{w_\sigma}^1, C_{w_\sigma}^2)$ to $\mathcal{A}_2$. Otherwise, $\mathcal{C}$ aborts the game (this event is denoted as Event_3)

Note that $C_{w_\sigma}^2$ is implicitly defined as $bQ_{\text{FS}} + bQ_{\text{BS}} + h_{2\sigma}Q_{\text{ch}} = b(T_{\text{FS}} + \beta_{\text{FS}}P_{\text{FS}} + \alpha_{\text{FS}}P_{\text{pub}}) + b(T_{\text{BS}} + \beta_{\text{BS}}P_{\text{BS}} + \alpha_{\text{BS}}P_{\text{pub}}) + h_{2\sigma}(T_{\text{ch}} + \beta_{\text{ch}}P_{\text{ch}} + \alpha_{\text{ch}}P_{\text{pub}})$.

(iv) *Phase*2: the oracles defined in phase 1 can be continuously accessed by $\mathcal{A}_2$

(v) *Guess*: $\mathcal{A}_2$ returns σ'. If $\sigma' = \sigma$, then $\mathcal{C}$ wins in the game

Table 1: Notions for some basic operations.

Symbols	Definition
T_{sm}	Runtime for a scalar multiplication in $\mathbb{G}$
T_{mm}	Runtime for a modular multiplication in $\mathbb{Z}_q^*$
T_{bp}	Runtime for a bilinear pairing
T_{pa}	Runtime for a point addition in $\mathbb{G}$
T_H	Runtime for a map-to-point hash
$\lvert\mathbb{G}\rvert$	The bit-size of a point in $\mathbb{G}$
$\lvert\mathbb{Z}_q\rvert$	The bit-size of a number in $\mathbb{Z}_q$
$\lvert\text{PK}\rvert$	The bit-size of PK
$\lvert C\rvert$	The bit-size of ciphertext
$\lvert T\rvert$	The bit-size of trapdoor

Table 2: The performance comparison.

Schemes	Computation cost (ms)				Communication cost (bit)		
	KeyGen	Encrypt	Trapdoor	Test	$\|PK\|$	$\|C\|$	$\|T\|$
Lu and Li [21]	$2T_{sm} + T_{mm}$	$3T_{sm} + T_{mm}$	T_{mm}	T_{sm}	$2\|\mathbb{G}\|$	$\|\mathbb{G}\| + \|\mathbb{Z}_q\|$	$\|\mathbb{Z}_q\|$
Peng et al. [18]	$8T_{sm} + 2T_H$	$4T_{sm} + 3T_H + 3T_{bp}$	$3T_{sm} + T_H$	$T_{sm} + 2T_{pa} + T_{bp}$	$4\|\mathbb{G}\|$	$\|\mathbb{G}\| + \|\mathbb{Z}_q\|$	$3\|\mathbb{G}\|$
DS-CLSPE	$6T_{sm}$	$10T_{sm} + 8T_{pa}$	$3T_{sm} + T_{pa}$	$4T_{sm} + 3T_{pa}$	$6\|\mathbb{G}\|$	$2\|\mathbb{G}\|$	$2\|\mathbb{G}\|$

At this point, $\mathscr{C}$ can compute abP as below: $Z = (t_{FS} + t_{BS})^{-1} \cdot (C_{w_\sigma}^2 - h_{2\sigma}(T_{ch} + \beta_{ch}P_{ch} + \alpha_{ch}P_{pub}) - (\beta_{FS}x_{FS} + \beta_{BS}x_{BS} + \alpha_{FS}s + \alpha_{BS}s)bP) = abP$.

Analysis. Now let us analyze $\mathscr{C}$'s advantage in winning the above game. $\mathscr{C}$ will win the game if Event_1, Event_2, and Event_3 do not occur.

From the above proof of Lemma 4,

$$\Pr\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3}\right] = \delta \cdot (1-\delta)^{q_{SK}+q_T}, \quad (23)$$

when $\delta = 1/(q_{SK} + q_T + 1)$, $\Pr\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3}\right]$ takes its maximum value

$$\begin{aligned}&\Pr\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3}\right] \\ &= \left(\frac{1}{q_{SK} + q_T + 1}\right)\left(1 - \frac{1}{q_{SK} + q_T + 1}\right)^{q_{SK}+q_T} \\ &\approx \frac{1}{(q_{SK} + q_T + 1)e}.\end{aligned} \quad (24)$$

Then, we have

$$\varepsilon' \geq \varepsilon \cdot \Pr\left[\overline{\text{Event}_1} \wedge \overline{\text{Event}_2} \wedge \overline{\text{Event}_3}\right] = \frac{\varepsilon}{(q_{SK} + q_T + 1)e}. \quad (25)$$

6. Performance Analysis

This section mainly compares the computation/communication costs of DS-CLSPE with that of Lu and Li [21] and Peng et al. [18]. Let p, q be 512-bit and 160-bit prime numbers, respectively. $\mathbb{G}$ is a cyclic group with order q, which is generated by a point on a super-singular elliptic curve $E(F_p)$. For the convenience of comparison, Table 1 presents the definition of some symbols.

We evaluate the running time of the above basic operations using the MIRACL library [34] and performing on a personal computer (Processor: i5-8250U 1.60 GHz; Memory: 8 GB; Operating system: Win10). The evaluation result shows that $T_{sm} = 4.800$ ms, $T_{mm} = 0.003$ ms, $T_{bp} = 13.144$ ms, $T_H = 12.082$ ms, $T_{pa} = 0.025$ ms. Furthermore, the result indicates that the operations of T_H and T_{bp} consume much more time than other operations. Therefore, we should minimize or even avoid using these time-consuming operations to enhance the efficiency of the designed scheme.

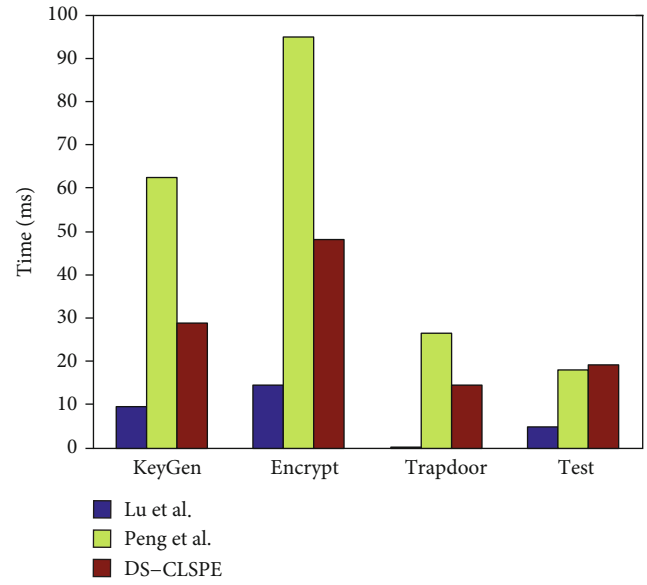

Figure 3: Computation cost comparison.

To compare the computation costs, we analyze the proposed DS-CLSPE scheme and schemes [18, 21] in terms of four phases: *KenGen*, *Encrypt*, *Trapdoor*, and *Test*. Table 2 and Figure 3 present the specific comparison results. In addition, the communication costs of DS-CLSPE and schemes [18, 21] are also presented in Table 2.

From Table 2 and Figure 3, the efficiency of DS-CLSPE is slightly worse than the scheme [21], but DS-CLSPE avoids the security flaws that existed in the scheme [21]. The data security is a primary concern in practical application, so DS-CLSPE is more practical. And in comparison with scheme [18], DS-CLSPE has better performance.

7. Conclusion

As the maturity of IoT and the popularization of sensor devices, IIoT has attracted widespread attention, which can provide users with real-time and reliable intelligent services by collecting and analyzing massive industrial data via the IoT devices. However, some sensitive information may be involved in industrial data, so data security is concerned. To protect data privacy, Lu et al. designed a CLSPE scheme without bilinear pairing operation. Unfortunately, we analyze that their CLSPE scheme cannot prevent user impersonation attacks. To resolve the security flaws, we design an improved pairing-free dual-server CLSPE scheme, i.e., DS-CLSPE. The formal security proof shows that DS-CLSPE

can realize IND-CKA security. Additionally, we evaluate the efficiency of DS-CLSPE, and evaluation results indicate the proposed scheme has better efficiency.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The work was supported by the National Natural Science Foundation of China (Nos. 61902111, 61932016, and 61972294), the High-level talent Fund Project of Henan University of Technology (No. 2018BS052), the Project funded by China Postdoctoral Science Foundation (No. 2020M670223), and the National Key Research and Development Program of China (No. 2018YFC1604000).

References

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): a vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] S. Pattar, R. Buyya, K. R. Venugopal, S. Iyengar, and L. Patnaik, "Searching for the iot resources: fundamentals, requirements, comprehensive review, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 2101–2132, 2018.

[3] S. Li, L. Da Xu, and S. Zhao, "5G internet of things: a survey," *Journal of Industrial Information Integration*, vol. 10, pp. 1–9, 2018.

[4] M. Younan, E. H. Houssein, M. Elhoseny, and A. A. Ali, "Challenges and recommended technologies for the industrial internet of things: a comprehensive review," *Measurement*, vol. 151, p. 107198, 2020.

[5] MarketsandMarkets, "Industrial iot market by device & technology (sensor, rfid, industrial robotics, dcs, condition monitoring, smart meter, camera system, networking technology), software (plm, mes, scada), vertical, and geography-global forecast to 2023," https://www.marketsandmarkets.com/Market-Reports/industrial-internet-of-things-market-129733727.html.

[6] M. S. Hossain and G. Muhammad, "Cloud-assisted industrial internet of things (IIoT) – enabled framework for health monitoring," *Computer Networks*, vol. 101, pp. 192–202, 2016.

[7] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 9, pp. 1981–1992, 2015.

[8] C. Esposito, A. De Santis, G. Tortora, H. Chang, and K.-K. R. Choo, "Blockchain: a panacea for healthcare cloud-based data security and privacy?," *IEEE Cloud Computing*, vol. 5, no. 1, pp. 31–37, 2018.

[9] J. Srinivas, A. K. Das, N. Kumar, and J. Rodrigues, "Cloud centric authentication for wearable healthcare monitoring system," *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 5, pp. 942–956, 2018.

[10] D. He, N. Kumar, S. Zeadally, and H. Wang, "Certificateless provable data possession scheme for cloud-based smart grid data management systems," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 3, pp. 1232–1241, 2018.

[11] A. K. Das, M. Wazid, N. Kumar, A. V. Vasilakos, and J. J. Rodrigues, "Biometrics-based privacy-preserving user authentication scheme for cloud-based industrial internet of things deployment," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4900–4913, 2018.

[12] X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, Berkeley, CA, USA, May 2000.

[13] D. Boneh, G. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004. EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027*, C. Cachin and J. L. Camenisch, Eds., pp. 506–522, Springer, Berlin, Heidelberg, 2004.

[14] 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," *Journal of Systems and Software*, vol. 83, no. 5, pp. 763–771, 2010.

[15] 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.

[16] M. Zeng, H.-F. Qian, J. Chen, and K. Zhang, "Forward secure public key encryption with keyword search for outsourced cloud storage," *IEEE Transactions on Cloud Computing*, 2019.

[17] 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. CRYPTO 2005. Lecture Notes in Computer Science, vol 3621*, V. Shoup, Ed., pp. 205–222, Springer, Berlin, Heidelberg, 2005.

[18] Y. Peng, J. Cui, C. Peng, and Z. Ying, "Certificateless public key encryption with keyword search," *China Communications*, vol. 11, no. 11, pp. 100–113, 2014.

[19] S. H. Islam, M. S. Obaidat, V. Rajeev, and R. Amin, "Design of a certificateless designated server based searchable public key encryption scheme," in *Mathematics and Computing. ICMC 2017. Communications in Computer and Information Science, vol 655*, D. Giri, R. Mohapatra, H. Begehr, and M. Obaidat, Eds., pp. 3–15, Springer, Singapore, 2017.

[20] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3618–3627, 2017.

[21] Y. Lu and J. Li, "Constructing pairing-free certificateless public key encryption with keyword search," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 8, pp. 1049–1060, 2019.

[22] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *2012 IEEE International Conference on Communications (ICC)*, pp. 917–922, Ottawa, ON, Canada, June 2012.

[23] C. Zuo, S.-F. Sun, J. K. Liu, J. Shao, and J. Pieprzyk, "Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security," in *Computer Security. ESORICS 2018. Lecture Notes in Computer Science,*

vol 11099, J. Lopez, J. Zhou, and M. Soriano, Eds., pp. 228–246, Springer, Cham, 2018.

[24] Z. Zhang, J. Wang, Y. Wang, Y. Su, and X. Chen, "Towards efficient verifiable forward secure searchable symmetric encryption," in *Computer Security – ESORICS 2019. ESORICS 2019. Lecture Notes in Computer Science, vol 11736*, K. Sako, S. Schneider, and P. Ryan, Eds., pp. 304–321, Springer, Cham, 2019.

[25] C. Lin, D. He, X. Huang, X. Xie, and K.-K. R. Choo, "Blockchain-based system for secure outsourcing of bilinear pairings," *Information Sciences*, vol. 527, pp. 590–601, 2020.

[26] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: the power of file-injection attacks on searchable encryption," in *Proceedings of the 25th Security Symposium, USENIX*, pp. 707–720, Austin, TX, USA, August 2016.

[27] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Advances in Cryptology. CRYPTO 1984. Lecture Notes in Computer Science, vol 196*, G. R. Blakley and D. Chaum, Eds., pp. 47–53, Springer, Berlin, Heidelberg, 1984.

[28] Y. Lu, G. Wang, J. Li, and J. Shen, "Efficient designated server identity-based encryption with conjunctive keyword search," *Annals of Telecommunications*, vol. 72, no. 5-6, pp. 359–370, 2017.

[29] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Information Sciences*, vol. 481, pp. 330–343, 2019.

[30] X. Zhang, Y. Tang, H. Wang, C. Xu, Y. Miao, and H. Cheng, "Lattice-based proxy-oriented identity-based encryption with keyword search for cloud storage," *Information Sciences*, vol. 494, pp. 193–207, 2019.

[31] S. S. Al-Riyami and K. G. Paterso, "Certificateless public key cryptography," in *Advances in Cryptology - ASIACRYPT 2003. ASIACRYPT 2003. Lecture Notes in Computer Science, vol 2894*, C. S. Laih, Ed., pp. 452–473, Springer, Berlin, Heidelberg, 2003.

[32] B. C. Hu, D. S. Wong, Z. Zhang, and X. Deng, "Certificateless signature: a new security model and an improved generic construction," *Designs, Codes and Cryptography*, vol. 42, no. 2, pp. 109–126, 2007.

[33] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 789–798, 2015.

[34] "Shamus software ltd., miracl library," http://www.shamus.ie/index.php?page=home.