Parallel Key-Insulated Multiuser Searchable Encryption for Industrial Internet of Things

Jie Cui, Jie Lu, Hong Zhong, Qingyang Zhang, Chengjie Gu, and Lu Liu

***Abstract*—With the rapid development of the industrial Internet of Things (IIoT) and cloud computing, an increasing number of companies outsource their data to cloud servers to save costs. To protect data privacy, sensitive industrial data must be encrypted before being outsourced to cloud servers. A multiuser searchable encryption (MUSE) scheme was introduced to ensure high efficiency of encrypted data retrieval. In an IIoT system with numerous users, the existing MUSE schemes suffer from certain key exposure problems owing to the limited key protection of smart devices and frequent queries by users. In this article, we propose a parallel key-insulated MUSE scheme for IIoT. This scheme utilizes broadcast encryption technology to implement MUSE. In addition, our scheme introduces a key-insulated primitive to improve the tolerance to key exposure. The security of our scheme is proved in the random oracle model. The experimental results show that our scheme achieves high computational efficiency.**

***Index Terms*—Industrial Internet of Things (IIoT), key exposure, key-insulated, multiuser (MU) searchable encryption (MUSE).**

I. Introduction

With the rise of "Industrial Revolution 4.0," the industrial Internet of Things (IIoT) [1], which is the result of the rapid development and application of the Internet of Things in the industrial field, has attracted widespread attention from society. In IIoT, industrial data are monitored, collected, exchanged, and analyzed by connecting various physical devices, such as sensors and actuators. The use of IIoT reduces the cost of production and consumption of resources, improves product quality and performance, and makes manufacturing and industrial processes more intelligent. In addition, with the rapid development of cloud computing via powerful data processing and storage capabilities, an increasing amount of IIoT data are outsourced to the cloud for storage. This reduces the cost of data management and improves the efficiency of industrial manufacturing.

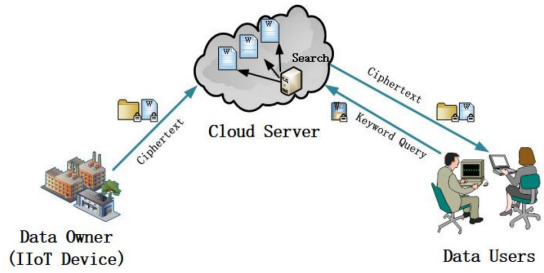


Fig. 1. MUSE scenario for IIoT.

Although data outsourcing presents several benefits to industrial management, it also introduces major issues concerning data security and privacy [2], [3] in cloud-based IIoT. The privacy of outsourced data depends entirely on the cloud servers (CSs). However, CSs are not completely trustworthy. To protect data privacy, sensitive industrial data must be encrypted before being outsourced to CSs. However, traditional encryption schemes make it difficult to retrieve data from the CSs. To overcome this challenge, multiple secure keyword searchable encryption (SE) schemes [4]–[8] have been proposed for effective ciphertext retrieval and data sharing.

In an IIoT system with numerous users, the data owner (DO) authorizes multiple users to share encrypted data and allows them to perform keyword queries on the shared data in the CS. This scenario is called multiuser (MU) searchable encryption (MUSE). Fig. 1 illustrates the MUSE scenario for IIoT. In an industrial production process, various physical devices collect large amounts of data. The DO encrypts these data and keywords, and uploads them to the CS via the Internet. Furthermore, data users (DUs) can send keyword queries to CSs. The CS verifies whether the query matches the keyword ciphertext and then returns the data ciphertext containing the keyword to the users.

However, there exists a key exposure problem in the deployment and application of IIoT. In a large-scale IIoT system, an increasing number of mobile smart terminal devices are used,

Manuscript received June 22, 2021; revised August 7, 2021; accepted August 25, 2021. Date of publication September 3, 2021; date of current version April 13, 2022. This work was supported in part by the National Natural Science Foundation of China under Grant 61872001, Grant 62011530046, and Grant U1936220 and in part by the Special Fund for Key Program of Science and Technology of Anhui Province, China under Grant 202003A05020043. Paper no. TII-21-2602. *(Corresponding author: Hong Zhong.)*

Jie Cui, Jie Lu, Hong Zhong, and Qingyang Zhang are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230039, China, with the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China, and also with the Institute of Physical Science and Information Technology, Anhui University, Hefei 230039, China (e-mail: cuijie@mail.ustc.edu.cn; xz2249182289@163.com; zhongh@ahu.edu.cn; qingyang.zhang.inchina@gmail.com).

Chengjie Gu is with Security Research Institute, New H3C Group, Hefei 230088, China (e-mail: gu.chengjie@h3c.com).

Lu Liu is with the School of Informatics, University of Leicester, LE1 7RH Leicester, U.K. (e-mail: l.liu@leicester.ac.uk).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2021.3110193.

Digital Object Identifier 10.1109/TII.2021.3110193

and the protection of keys by these devices is limited. DUs often use their keys to perform query operations on these insecure devices and then submit queries to the CS. In this process, a malicious adversary can easily steal the user's key information, which can lead to key exposure problems. Once the user's private key is compromised, the adversary may use the exposed private key to submit a legitimate request to the CS. The CS successfully verifies it and returns the previously encrypted data to the adversary. Eventually, the adversary can use the exposed private key to decrypt the encrypted data. This is a significant hazard.

Some existing SE schemes [9]–[12] reduce the hazard of key exposure through supporting forward security (FS). However, these schemes can only protect past keys and cannot protect future keys. The introduction of a key insulation primitive [13], [14] can effectively address the key exposure problem. The idea of key insulation is to store long-term keys in a physically secure but computationally limited device called a helper. The short-term key is stored on a powerful but insecure device, which is updated through the helper every given time period. The key insulation scheme ensures that keys before and after that exposure time period cannot be deduced from the already exposed key, i.e., forward and backward security (BS) is guaranteed. The MUSE scenarios with numerous users are more vulnerable to key exposure, but few MUSE schemes consider this security issue.

Another common problem in data sharing is that DOs selectively share their data with the users. To protect the privacy of shared data, the DO needs to use different keys to encrypt different files, which is called multikey searchable encryption (MKSE). Most existing MKSE schemes [15], [16] do not implement access control (AC). Any user can search and decrypt files, which may lead to a breach in the data privacy.

For the abovementioned issues, it is a challenge for IIoT to design a MUSE that can solve the key exposure problem and support multikey encryption (MK).

A. Related Work

An increasing number of individuals and enterprises store large amounts of industrial data in the cloud, multiple SE schemes [17], [18] have been proposed to ensure efficient ciphertext retrieval.

Single-user SE. SE was first realized through symmetric encryption, called symmetric searchable encryption (SSE), proposed by Song *et al.* [19]. SSE allows the CS to perform searches while protecting privacy, but it has a high key management overhead in symmetric settings. To solve this problem, Boneh *et al.* [20] first proposed the public-key encryption with keyword search (PEKS) scheme. The DO encrypts the data and keywords with the public key of the target receiver. The data receiver can use his private key to generate a query trapdoor and submit it to the CS to retrieve the ciphertext. Subsequently, many variants of the PEKS were proposed.

Tian *et al.* [21] proposed an identity-based PEKS scheme to simplify the management of public key and certificate in traditional PEKS based on public-key infrastructure (PKI). He *et al.* [22] and Ma *et al.* [23] proposed a certificate-less PEKS scheme for IIoT that solves the key escrow problem, respectively. However, most PEKS schemes include expensive bilinear pairing operations and modular exponentiation operations, it is difficult to solve the ciphertext retrieval problem in the MU scenario.

MUSE. In MUSE, the DO can encrypt the same keyword for a group of users to avoid unnecessary data redundancy. In the practical application of IIoT, the MU scenario is more common.

Attrapadung *et al.* [24] introduced an encryption primitive called Hierarchical Identity Coupled Broadcast Encryption, and constructed a public broadcast SE scheme. Then, Ali *et al.* [25] proposed a broadcast searchable keyword encryption (BSKE) scheme. In the scheme, the size of the ciphertext is fixed and does not increase with the number of users. Kiayias *et al.* [26] proposed a more effective MUSE scheme based on previous research. This scheme implements MK, and users can decrypt the decryption keys of the search results. But it is expensive because each user's key has 14 grouping elements. Lu *et al.* [27] proposed a certificate-less PEKS scheme for multiple users. The scheme reduces a lot of computational costs because it does not use bilinear pairing operations. However, the computation cost will increase with the number of users.

Key exposure problem. In the deployment and application of IIoT, how to ensure the security of user keys is a major challenge. In an IIoT system, numerous physical devices have limited protection of keys. Users use their keys to frequently perform queries on such insecure devices, key exposure may occur. Dodis *et al.* [13] first proposed the concept of key-insulated. By frequently updating the private key, the tolerance to key exposure can be improved. But this also means frequent connections between help devices (HDs) and unsecured networks, which increases the risk of help keys being exposed. Therefore, Hanaoka *et al.* [28] proposed the parallel key-insulated public key encryption scheme, which uses two secure physical devices as helpers to update keys alternately. They aim to reduce the risk of helpers' key exposure by reducing the frequency of their connections to insecure environments.

SE allows users in different geographical locations to share data, but inevitably suffers from the problem of key exposure. Recently, some SSE schemes [9], [10] have addressed this security issue by supporting FS. Later, the scheme proposed by Zhang *et al.* [11] uses the lattice basis delegation mechanism to achieve the FS of the system. Recently, Kim *et al.* [12] proposed a forward-secure PEKS scheme based on hierarchical identity-based encryption. However, few MUSE schemes address the problem of key exposure in previous studies.

B. Our Contribution

To solve the abovementioned problems, we propose a parallel key-insulated MUSE (PKI-MUSE) scheme for IIoT. First, the PKI-MUSE scheme is based on the BSKE scheme [25] to realize MUSE. Second, our scheme integrates the key-insulated primitive and improves the tolerance to key exposure. In addition, the parallel mechanism allows frequent updates of the user key while reducing the opportunity to expose the helper

key, thus improving the security of the system. Specifically, our contributions are as follows.

1) The PKI-MUSE scheme integrates the key-insulated primitive to improve the tolerance to key exposure through user key updates in each time period. In addition, the user key update does not require re-encrypting the data, which significantly reduces the encryption overhead.
2) The PKI-MUSE scheme combines MK and ac to improve the efficiency and security of search and decryption.
3) We provide security proof of our PKI-MUSE scheme in the random oracle model [29]. In addition, through the experimental simulation, we show the high efficiency and practicability of our scheme.

C. Organization

The rest of this article is organized as follows. Section II introduces the preliminaries. Section III describes the scheme model and its security definition. In Section IV, we describe the proposed scheme and prove its security in Section V. Section VI provides some performance evaluations. Finally, Section VII concludes this article.

II. Preliminaries

In this section, we introduce some preliminaries, including the properties of bilinear mapping and the security assumptions of the proposed scheme.

A. Bilinear Mapping

Let G_1 and G_2 be additive cyclic groups of order p, G_T be a multiplicative group of order p. p is a prime number, g and h are generators of G_1 and G_2, respectively. A bilinear mapping $e : G_1 \times G_2 \to G_T$ must satisfy the following properties:

1) bilinearity: Given any $a, b \in Z_p^*$, $e(g^a, h^b) = e(g, h)^{ab} \in G_T$;
2) nondegenerate: $e(g, h) \neq 1$;
3) computability: There is polynomial time algorithm by given any $g \in G_1$, $h \in G_2$ that can calculate $e(g, h) \in G_T$.

B. Security Assumptions

We present a bilinear Diffie–Hellman exponent ((l, l)-BDHE) problem [30].

Let l be an integer and $e : G_1 \times G_2 \to G_T$ be a bilinear mapping, where G_1 and G_2 be additive cyclic groups of prime order p, G_T be a multiplicative group of order p. Given $3l + 2$ elements $(v, g, h, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^l}, g^{\alpha^{l+2}}, \ldots, g^{\alpha^{2l}}, h^{\alpha}, h^{\alpha^2}, \ldots, h^{\alpha^l})$ as input, output vector $e(g, v)^{\alpha^{l+1}} \in G_T$, where $g, g^{\alpha^i} \in G_1$ and $v, h, h^{\alpha^i} \in G_2$.

For convenience, let $g_i = g^{\alpha^i}$, $h_i = h^{\alpha^i}$. Let $\mathcal{A}$ be a τ-time algorithm that takes an input challenge for (l, l)-BDHE and outputs a decision bit $b \in \{0, 1\}$. We say that $\mathcal{A}$ has an advantage ε to solve problem (l, l)-BDHE if

$$\Pr[\mathcal{A}(v, g, h, g_1, g_2, \ldots, g_l, g_{l+2}, \ldots, g_{2\,l}, h_1, h_2, \ldots, h_l) = e(g_{l+1}, v)] \geq \varepsilon \tag{1}$$

where probability is distributed over a random selection of $g \in G_1$ and $v, h \in G_2$, random choice of $\alpha \in Z_p^*$, random of choice of $T \in G_T$, and random bits selected by $\mathcal{A}$.

Definition 1: The $(\tau, \varepsilon, l, l)$-BDHE assumption holds in (G_1, G_2) if no τ-time algorithm has advantage at least ε in solving the (l, l)-BDHE problem in (G_1, G_2).

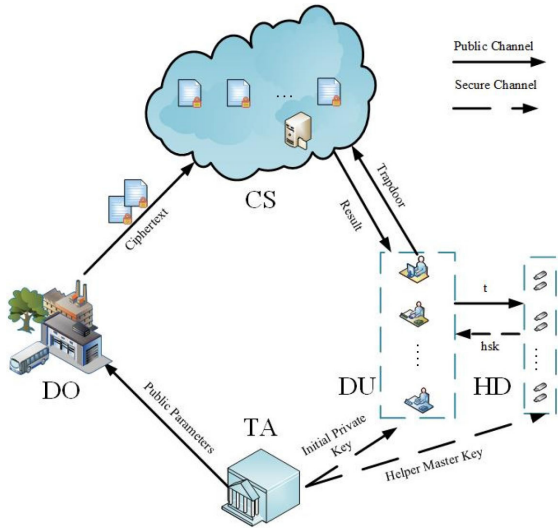


Fig. 2. System model of PKI-MUSE.

III. System Model and Security Definition

In this section, we present the system model and security definition of the PKI-MUSE scheme.

A. System Model

As shown in Fig. 2, the scheme consists of the following five entities, namely: a trusted authority (TA), a CS, a DO, multiple DUs, and two HD for each DU.

TA: The TA is responsible for generating a set of system parameters and a system master key. At the same time, it is responsible for distributing an initial key to each DU and two helper master keys to each HD.

DO: The DO is responsible for encrypting IIoT data and keywords, and setting a subset of authorized users, then uploading the ciphertext and subset to the CS. It is also responsible for adding and removing users.

DU: The DU is responsible for generating trapdoors for the keywords he/she wants to search for and sending them to the CS for query requests.

HD: The HD is responsible for generating a helper key to update the private key of the DU.

CS: The CS locates all the matching ciphertext using the search trapdoor and returns them to the user. In our scheme, the TA will first run the setup, generate the system master key mst,

public parameter param, and the helper master key $\{s_{i,0}, s_{i,1}\}$. Then, the TA runs the KeyGen to generate the initial key $\text{usk}_{i,0}$, sends $\text{usk}_{i,0}$ to each user and sends $\{s_{i,0}, s_{i,1}\}$ to each HD, respectively. Next, the DO runs the encrypt to generate ciphertext C and subset S and sends them to the CS. The DU runs the trapdoor to generate a trapdoor and submits it to the CS for a query request. Finally, the CS runs the search and sends all matching ciphertexts to the user. The DU can run the decrypt to get data. If necessary, the DU will run the update to update his private key with the HD's key.

B. Scheme Framework

The PKI-MUSE scheme is as follows.

$\text{Setup}(\lambda, n) \to \{\text{mst}, s_{i,0}, s_{i,1}, \text{param}\}$: Input a security parameter λ and the maximum number of users n, return the system master key mst, and the public parameter param and generate the helper master key $\{s_{i,0}, s_{i,1}\}$, where $i \in \{1, 2, \ldots, n\}$.

$\text{KeyGen}(i, \text{mst}, s_{i,0}, s_{i,1}) \to \{\text{usk}_{i,0}\}$: TA generates an initial key $\text{usk}_{i,0}$ for user i and sends $\text{usk}_{i,0}$ to user i through a secure channel. Then TA sends $\{s_{i,0}, s_{i,1}\}$ to the two HDs of user i through a secure channel, respectively.

$\text{Update}(t, \text{usk}_{i,t-1}, s_{i,k}) \to \{\text{usk}_{i,t}\}$: The DU inputs time period t, where $t \in \{1, 2, \ldots, N\}$. The HD $k(k \equiv t \pmod 2)$ updates helper key $\text{hsk}_{i,t}$ and sends to the user. The user then updates the key $\text{usk}_{i,t}$ for the period t.

$\text{Encrypt}(F, W) \to \{C, S\}$: The DO selects different symmetric keys to encrypt files $F = \{f_1, f_2, \ldots, f_m\}$, generates file ciphertext C_f and extractes keywords $W = \{w_1, w_2, \ldots, w_q\}$ to generate keyword ciphertext C_w. Set up a subset of authorized users $S \subseteq \{1, 2, \ldots, n\}$ and upload $C = (C_f, C_w)$ and S to the CS for storage.

$\text{Trapdoor}(t, \text{usk}_{i,t}, w', S) \to \{T_{i,t}\}$: The DU uses the key $\text{usk}_{i,t}$ and the keyword w' that you want to search to generate trapdoor $T_{i,t}$ and submits it to the CS for query requests.

$\text{Search}(i, t, C, T_{i,t}, S) \to \{C'\}$: The CS first checks whether the user i is valid (included in S), then verify whether the keyword ciphertext matches the query trapdoor. Finally, the server sends all matching ciphertext C' to the user.

$\text{Decrypt}(\text{usk}_{i,t}, C') \to \{F'\}$: The user uses $\text{usk}_{i,t}$ to decrypt the symmetric key and then uses the symmetric key to decrypt the ciphertext to obtain the files F'.

$\text{AddUser}(x)$: When adding a user x, the DO first adds user x to the authorized user subset S and then modifies C_2 in the keyword ciphertext C_w.

$\text{RevokeUser}(x)$: When revoking a user x, the DO first deletes the user x from the authorized user subset S and then modifies the C_2 in the keyword ciphertext C_w.

C. Security Definition

Here, we define semantic security for the PKI-MUSE scheme. This is based on the security definition of [13] and [25].

Setup: Challenger $\mathcal{C}$ runs the setup to generate param, mst and $\{s_{i,0}, s_{i,1}\}$. He sends param to adversary $\mathcal{A}$, keeping mst and $\{s_{i,0}, s_{i,1}\}$ only known by himself.

Phase 1: Adversary $\mathcal{A}$ publishes the following series of queries.

Exposure queries $\langle i, t, \text{class}\rangle$: If class = user, $\mathcal{C}$ runs the KeyGen and the update and gets a temporary key $\text{usk}_{i,t}$, then returns it to $\mathcal{A}$. If class = helper, $\mathcal{C}$ sends the helper key $s_{i,k}(k \equiv t \pmod 2)$ to $\mathcal{A}$.

Trapdoor queries $\langle i, t, w, S\rangle$: $\mathcal{C}$ runs the KeyGen and the update to obtain a temporary key $\text{usk}_{i,t}$. Then, he runs the trapdoor to obtain a trapdoor $T_{i,t}$ and returns it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ selects two challenge keywords of the same length w_0, w_1 and time period $t^* \in \{1, 2 \ldots, N\}$ and sends it to challenger $\mathcal{C}$. The challenger randomly selects $b \in \{0, 1\}$ and generates the challenge ciphertext C_b with the keyword w_b, then sends it to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ makes the second round of exposure queries and trapdoor queries as in phase 1. Note that in phase 1 and phase 2, challenge user $i \notin S$ and $\langle i, t^*, w_b, S\rangle$ cannot appear in trapdoor query lists.

Guess: $\mathcal{A}$ runs the search to do a test. Finally $\mathcal{A}$ sends a guess $b' \in \{0, 1\}$ to the challenger. $\mathcal{A}$ wins the game if $b = b'$.

Let us define the advantage of $\mathcal{A}$

$$\text{Adv}^{\text{PKI-MUSE}}(\mathcal{A}) = \Pr[b = b'] - \frac{1}{2}. \tag{2}$$

IV. Proposed PKI-MUSE Scheme

In this section, we show the specific construction of the PKI-MUSE scheme.

A. System Setup

$\text{Setup}(\lambda, n) \to \{\text{mst}, s_{i,0}, s_{i,1}, \text{param}\}$: Input a security parameter λ and the maximum number of users n, then generate a bilinear set of parameters (p, G_1, G_2, G_T, e). The TA chooses two generators $g \in G_1$, $h \in G_2$, a random number $\alpha \in Z_p^*$, and calculates $g_i = g^{\alpha^i}$, $h_i = h^{\alpha^i}$ for $i = (1, 2, \ldots, n, n+2, \ldots, 2n)$. Then, the TA chooses a random number $\gamma \in Z_p^*$ and generates the helper master keys $\{s_{i,0}, s_{i,1}\}$ for the two HDs of user i, respectively, where $i \in \{1, 2, \ldots, n\}$. Calculate system public key $\text{pk} = g^\gamma$, the public key of each HD $\text{pk}_{i,0} = h^{s_{i,0}}$, $\text{pk}_{i,1} = h^{s_{i,1}}$. Choose two collision resistant hash functions $H_1 : \{0, 1\}^* \to Z_p^*$ and $H_2 : \{0, 1\}^* \to G_1$, then calculate $u_{-1} = g^{H_1(-1)}$, $u_0 = g^{H_1(0)}$. Finally, the TA chooses a pair of semantically secure symmetric encryption and decryption algorithms E and D. The public parameter and system master key are $\text{param} = \{\{g_i\}_{(i=1,2,\ldots,n,n+2,\ldots,2n)}, \{h_i\}_{(i=1,2,\ldots,n,n+2,\ldots,2n)}, \text{pk}, \text{pk}_{i,0}, \text{pk}_{i,1}, H_1, H_2, u_{-1}, u_0\}$, and $\text{mst} = \gamma$, respectively.

B. Key Generation

$\text{KeyGen}(i, \text{mst}, s_{i,0}, s_{i,1}) \to \{\text{usk}_{i,0}\}$: The TA first calculates $d_{i,-1} = u_{-1}^{s_{i,1}}$, $d_{i,0} = u_0^{s_{i,0}}$ and generates an initial key $\text{usk}_{i,0} = (g_i)^\gamma d_{i,-1} d_{i,0}$. Then, the TA sends $\text{usk}_{i,0}$ to user i through a secure channel and sends $\{s_{i,0}, s_{i,1}\}$ to the two HDs of user i through a secure channel, respectively.

C. Key Update

$\text{Update}(t, \text{usk}_{i,t-1}, s_{i,k}) \to \{\text{usk}_{i,t}\}$: Input a time period t, where $t \in \{1, 2, \ldots, N\}$, the HD $k(k \equiv t \pmod 2)$

calculates $u_{t-2} = g^{H_1(t-2)}$, $d_{i,t-2} = u_{t-2}^{s_{i,k}}$, $d_{i,t} = u_t^{s_{i,k}}$. Then, the HD $k(k \equiv t \pmod 2)$ updates helper key $\text{hsk}_{i,t} = d_{i,t-2}^{-1} d_{i,t}$ and sends to the user. Finally, the user calculates $\text{usk}_{i,t} = \text{usk}_{i,t-1}\text{hsk}_{i,t}$, deletes $\text{usk}_{i,t-1}$ and $\text{hsk}_{i,t}$.

D. Data Encryption

Encrypt$(F, W) \to \{C, S\}$: For files $F = \{f_1, f_2, \ldots, f_m\}$, the DO first selects different symmetric keys K_x, calculates file ciphertext $C_x = E(K_x, f_x)$. Then, the DO extracts the keywords $W = \{w_1, w_2, \ldots, w_q\}$ and sets authorized user subset $S \subseteq \{1, 2, \ldots, n\}$ to access files F. The DO selects a random number $r \in Z_p^*$ and calculates $C_0 = h^r$, $C_1 = h_1^r$, $C_2 = (\text{pk}\prod_{j\in S} g_{n+1-j})^r$, $C_{3,y} = H_2(w_y)^r$, $C_4 = u_{-1}^r$, $C_5 = u_0^r$, $C_{f,x} = K_x e(g_1, h_n)^r$. Finally, the DO uploads ciphertext $C = (C_f, C_w)$ to the cloud storage and publishes S to the authorized users and the CS, where $C_f = (C_x, C_{f,x})$, $C_w = (C_0, C_1, C_2, C_{3,y}, C_4, C_5)$.

E. Trapdoor Generation

Trapdoor$(t, \text{usk}_{i,t}, w', S) \to \{T_{i,t}\}$: The DU selects a random number $z \in Z_p^*$ and generates the trapdoor $T_{i,t}$ in time period t by calculating $T_0 = h^z$, $T_1 = h_i^z$, $T_2 = g_n^z$, $T_3 = (\text{usk}_{i,t} H_2(w') \prod_{j\in S, j\neq i} g_{n+1-j+i})^z$, $T_4 = \text{pk}_{i,0}^z$, $T_5 = \text{pk}_{i,1}^z$. Then, the DU sends $T_{i,t} = (T_0, T_1, T_2, T_3, T_4, T_5)$ to the CS for a query.

F. Search

Search$(i, t, C, T_{i,t}, S) \to \{C'\}$: The CS first checks whether user i is valid (included in S) and then calculates:

If $t \equiv 0 \pmod 2$

$$K = e(T_4, C_5)^{H_1(t)/H_1(0)} e(T_5, C_4)^{H_1(t-1)/H_1(-1)}. \quad (3)$$

Otherwise $t \equiv 1 \pmod 2$

$$K = e(T_4, C_4)^{H_1(t-1)/H_1(-1)}) e(T_5, C_5)^{H_1(t)/H_1(0)}. \quad (4)$$

They verify whether (5) holds. If the equation holds, output "true," the CS sends the matching ciphertext C' to the user, otherwise output "false." Note that $C' \subseteq C$

$$K \stackrel{?}{=} \frac{e(T_3, C_0) e(T_2, C_1)}{e(T_1, C_2) e(C_{3,y}, T_0)}. \quad (5)$$

G. Data Decryption

Decrypt$(\text{usk}_{i,t}, C') \to \{F'\}$: After receiving C', the DU calculates

If $t \equiv 0 \pmod 2$

$$K' = e(\text{pk}_{i,0}, C_5)^{H_1(t)/H_1(0)} e(\text{pk}_{i,1}, C_4)^{H_1(t-1)/H_1(-1)}. \quad (6)$$

Otherwise $t \equiv 1 \pmod 2$

$$K' = e(\text{pk}_{i,0}, C_4)^{H_1(t-1)/H_1(-1)} e(\text{pk}_{i,1}, C_5)^{H_1(t)/H_1(0)}. \quad (7)$$

Obtain the symmetric key K_x by (8), then the user decrypts the file $f_x = D(K_x, C_x)$. Where $\text{pub} = \prod_{j\in S, j\neq i} g_{n+1-j+i}$. Note that the pub of the set S can only be calculated once for efficiency

$$K_x = \frac{e(\text{usk}_{i,t}\text{pub}, C_0) C_{f,x}}{e(h_i, C_2) K'}. \quad (8)$$

H. User Addition and Revocation

AddUser(x): When adding a user x, the DO first adds user x to the authorized user subset S and then modifies $C_2 = C_2 \cdot (g_{n+1-x})^r$ in the keyword ciphertext C_w.

RevokeUser(x): When revoking a user x, the DO first deletes the user x from the authorized user subset S and then modifies $C_2 = C_2/(g_{n+1-x})^r$ in the keyword ciphertext C_w.

V. Security Proof

In this section, we present the correctness verification and security proof of the scheme.

A. Correctness Verification

We first verify the search. For (3) and (4), we expand the calculations in (9) and (10), then verify the correctness of (5) in (11).

If $t \equiv 0 \pmod 2$

$$\begin{aligned} K &= e(T_4, C_5)^{H_1(t)/H_1(0)} e(T_5, C_4)^{H_1(t-1)/H_1(-1)} \\ &= e(\text{pk}_{i,0}^z, u_0^r)^{H_1(t)/H_1(0)} e(\text{pk}_{i,1}^z, u_{-1}^r)^{H_1(t-1))/H_1(-1)} \\ &= e(\text{pk}_{i,0}^z, u_t^r) e(\text{pk}_{i,1}^z, u_{t-1}^r). \end{aligned} \quad (9)$$

Otherwise $t \equiv 1 \pmod 2$

$$\begin{aligned} K &= e(T_4, C_4)^{H_1(t-1)/H_1(-1)}) e(T_5, C_5)^{H_1(t)/H_1(0)} \\ &= e(\text{pk}_{i,0}^z, u_{-1}^r)^{H_1(t-1)/H_1(-1)} e(\text{pk}_{i,1}^z, u_0^r)^{H_1(t)/H_1(0)} \\ &= e(\text{pk}_{i,0}^z, u_{t-1}^r) e(\text{pk}_{i,1}^z, u_t^r) \end{aligned} \quad (10)$$

$$\begin{aligned} &\frac{e(T_3, C_0) e(T_2, C_1)}{e(T_1, C_2) e(C_{3,y}, T_0)} \\ &= \frac{e((\text{usk}_{i,t} H_2(w') \prod_{j\in S, j\neq i} g_{n+1-j+i})^z, h^r) e(g_n^z, h_1^r)}{e(h_i^z, (\text{pk}\prod_{j\in S} g_{n+1-j})^r) e(H_2(w_y)^r, h^z)} \\ &= \frac{e((\text{usk}_{i,t} H_2(w'))^z, h^r) e(\prod_{j\in S} g_{n+1-j+i}, h^r)^z e(g_n^z, h_1^r)}{e(h_i^z, (\text{pk}\prod_{j\in S} g_{n+1-j})^r) e(H_2(w_y)^r, h^z) e(g_{n+1}^z, h^r)} \\ &= \frac{e(\text{usk}_{i,t}, h^r)^z e(H_2(w'), h^r)^z e(\prod_{j\in S} g_{n+1-j+i}, h^r)^z}{e(h_i^z, \text{pk})^r e(h_i^z, \prod_{j\in S} g_{n+1-j})^r e(H_2(w_y)^r, h^z)} \\ &= \frac{e(g_i^\gamma, h^r)^z e(d_{i,t-1}, h^r)^z e(d_{i,t}, h^r)^z}{e(h_i^z, g^\gamma)^r} \\ &= e(u_{t-1}^{s_{i,k}}, h^r)^z e(u_t^{s_{i,k}}, h^r)^z \\ &= e(u_{t-1}^r, h^{s_{i,k}})^z e(u_t^r, h^{s_{i,k}})^z \\ &= K. \end{aligned} \quad (11)$$

Then. we verify the decrypt. For (6) and (7), we expand the calculations in (12) and (13), then verify the correctness of (8) in (14).

If $t \equiv 0 \pmod 2$

$$\begin{aligned} K' &= e(\mathrm{pk}_{i,0}, C_5)^{H_1(t)/H_1(0)} e(\mathrm{pk}_{i,1}, C_4)^{H_1(t-1)/H_1(-1)} \\ &= e(\mathrm{pk}_{i,0}, u_0^r)^{H_1(t)/H_1(0)} e(\mathrm{pk}_{i,1}, u_{-1}^r)^{H_1(t-1)/H_1(-1)} \\ &= e(\mathrm{pk}_{i,0}, u_t^r) e(\mathrm{pk}_{i,1}, u_{t-1}^r). \end{aligned} \tag{12}$$

Otherwise $t \equiv 1 \pmod 2$

$$\begin{aligned} K' &= e(\mathrm{pk}_{i,0}, C_4)^{H_1(t-1)/H_1(-1)} e(\mathrm{pk}_{i,1}, C_5)^{H_1(t)/H_1(0)} \\ &= e(\mathrm{pk}_{i,0}, u_{-1}^r)^{H_1(t-1)/H_1(-1)} e(\mathrm{pk}_{i,1}, u_0^r)^{H_1(t)/H_1(0)} \\ &= e(\mathrm{pk}_{i,0}, u_{t-1}^r) \cdot e(\mathrm{pk}_{i,1}, u_t^r) \end{aligned} \tag{13}$$

$$\begin{aligned} &\frac{e(\mathrm{usk}_{i,t} pub, C_0) C_{f,x}}{e(h_i, C_2) K'} \\ &= \frac{e(g_i^\gamma d_{i,t-1} d_{i,t}, h^r) e(\prod_{j\in S, j\neq i} g_{n+1-j+i}, h^r) C_{f,x}}{e(h_i, (\mathrm{pk} \prod_{j\in S} g_{n+1-j})^r) K'} \\ &= \frac{e(g_i^\gamma d_{i,t-1} d_{i,t}, h^r) e(\prod_{j\in S} g_{n+1-j+i}, h^r) K_x e(h_n, g_1)^r}{e(h_i, g^\gamma)^r e(h_i, \prod_{j\in S} g_{n+1-j})^r e(g_{n+1}, h^r) K'} \\ &= \frac{e(g_i^\gamma, h^r) e(d_{i,t-1}, h^r) e(d_{i,t}, h^r) K_x e(h_n, g_1)^r}{e(h_i, g^\gamma)^r e(g_{n+1}, h^r) K'} \\ &= \frac{e(d_{i,t-1}, h^r) e(d_{i,t}, h^r) K_x e(h_n, g_1)^r}{e(g_{n+1}, h^r) K'} \\ &= K_x. \end{aligned} \tag{14}$$

We can clearly see that if $w_y = w'$, our scheme is correct and users can successfully get the searched files.

B. Security Proof

Suppose there is an adversary $\mathcal{A}$ with an advantage ε to destroy the (l, l)-BDHE problem. We set up a challenger $\mathcal{C}$, which has a running time close to $\mathcal{A}$s running time.

Here, we assume that the adversary does not require exposure queries $\langle i, t, \text{helper}\rangle$ for any period t.

Challenger $\mathcal{C}$ responds to adversary $\mathcal{A}$s queries as follows.

Setup: To generate param, challenger $\mathcal{C}$ randomly selects $\beta \in Z_p^*$, calculates

$$\mathrm{pk} = g^\beta \cdot \left(\prod_{j\in S} g_{n+1-j} \right)^{-1} \tag{15}$$

$$\begin{aligned} \mathrm{param} = \{ &\{g_i\}_{(i=1,2,\ldots,n,n+2,\ldots,2n)}, \mathrm{pk}, \mathrm{pk}_{i,0}, \mathrm{pk}_{i,1} \\ &\{h_i\}_{(i=1,2,\ldots,n,n+2,\ldots,2n)}, H_1, H_2 \} \end{aligned} \tag{16}$$

and gives param to $\mathcal{A}$.

H1-queries: $\mathcal{A}$ publishes q_{H1} H1-queries. $\mathcal{C}$ prepares a list of tuples $\langle t, z\rangle$ to simulate the H1 function, called H1-list. The list is initially empty. When $\mathcal{A}$ asks a query t to challenger.

1) Search the entire H1-list and return $H_1(t) = z$ to $\mathcal{A}$ when the query t is in the H1-list.
2) Otherwise, the challenger randomly selects $z \in Z_p^*$, returns $H_1(t) = z$ to $\mathcal{A}$ and adds the tuple $\langle t, z\rangle$ to the H1-list.

H2-queries: $\mathcal{A}$ publishes q_{H2} H2-queries. $\mathcal{C}$ prepares a list of tuples $\langle w_j, h_j, x_j, y_j\rangle$ to simulate the H2 function, called H2-list. When $\mathcal{A}$ asks user i for the hash value of keyword w, $\mathcal{C}$ responds as follows.

1) Search the entire H2-list and return $H_2(w) = h_i$ to $\mathcal{A}$ if w_i is found.
2) Otherwise, $\mathcal{C}$ randomly selects (u_i, x_i), $u_i \in \{0, 1\}$, $x_i \in Z_p^*$. If $u_i = 0$, calculates

$$h_i = g^{x_i} \prod_{j\in S} g_{n+1-j}. \tag{17}$$

Otherwise $u_i = 1$, calculates

$$h_i = (g^{x_i})^{\alpha^{y_i}} \prod_{j\in S} g_{n+1-j} \tag{18}$$

where $y_i \in Z_p^*$.

3) Add the tuple $\langle w_i, h_i, x_i, y_i\rangle$ to the H2-list and return $H_2(w) = h_i$ to $\mathcal{A}$.

Exposure queries phase 1: $\mathcal{A}$ publishes q_E exposure queries. When $\mathcal{A}$ sends a query $\langle i, t, \text{class}\rangle$ to the challenger.

1) If class = helper, terminates the query.
2) Otherwise for $i \notin S$, $\mathcal{C}$ runs H1-query and gets $H_1(t)$ from the H1-list, then calculates the temporary key

$$\mathrm{usk}_{i,t} = (g_i)^\beta d_{i,t-1} d_{i,t} (\prod_{j\in S} g_{n+1-j+i})^{-1}. \tag{19}$$

Trapdoor queries phase 1: $\mathcal{A}$ publishes q_T trapdoor queries. The adversary sends the trapdoor query of keyword w to the challenger, where user $i (i \notin S)$ private key is $\mathrm{usk}_{i,t}$, then challenger will respond as follows.

1) $\mathcal{C}$ runs H2-query and gets $H_2(w)$ from the H2-list. If $u_i = 1$, terminates the query.
2) Otherwise, when $u_i = 0$, $\mathcal{C}$ calculates

$$h_i = g^{x_i} \prod_{j\in S} g_{n+1-j} = H_2(w). \tag{20}$$

The challenger randomly selects $s \in Z_p^*$ and calculates $T_0 = (h^s)^{\alpha^{i-y_i}}$, $T_1 = h_i^s$, $T_2 = g_n^s$, $T_3 = (\mathrm{usk}_{i,t}(H_2(w))^{\alpha^i} \prod_{j\in S, j\neq i} g_{n+1-j+i})^s$, $T_4 = \mathrm{pk}_{i,0}^s$, $T_5 = \mathrm{pk}_{i,1}^s$. Then, $\mathcal{C}$ returns $T_{i,t} = (T_0, T_1, T_2, T_3, T_4, T_5)$ to $\mathcal{A}$.

Challenge: $\mathcal{A}$ selects two challenge keywords of the same length w_0, w_1 and period $t^* \in \{1, 2\ldots, N\}$ and sends to $\mathcal{C}$. $\mathcal{C}$ responds as follows.

1) $\mathcal{C}$ runs the H2-query twice and gets the values h_0, h_1 of $H_2(w_0)$ and $H_2(w_1)$ from the H2-list. For $i = 0, 1$, $\langle w_i, h_i, x_i, y_i\rangle$ is the corresponding tuple in H2-list.
2) If $u_0 = 0$ and $u_1 = 0$, the query terminates.
3) If $u_0 = 1$ and $u_1 = 1$, $\mathcal{C}$ randomly selects $b \in \{0, 1\}$ to select the values of $H_2(w_0)$ and $H_2(w_1)$.
4) Otherwise, the values of $H_2(w_0)$ and $H_2(w_1)$ are selected based on u_0 and u_1.
5) The challenger randomly selects $r \in Z_p^*$, then calculates $C_0 = h^r$, $C_1 = h_1^r$, $C_2 = (\mathrm{pk} \prod_{j\in S} g_{n+1-j})^r = g^\beta$, $C_3 = H_2(w_b)^r$, $C_4 = u_{-1}^r$, $C_5 = u_0^r$. Then $\mathcal{C}$ returns $C_w = (C_0, C_1, C_2, C_3, C_4, C_5)$ to $\mathcal{A}$.

TABLE I
FUNCTION COMPARISON

Scheme	MU	FS	BS	AC	MK
LFS-PEKS [11]	No	Yes	No	No	No
FS-PEKS [12]	No	Yes	No	No	No
BSKE [25]	Yes	No	No	Yes	No
SEMEKS [26]	Yes	No	No	Yes	Yes
MRCLKS [27]	Yes	No	No	No	No
Our Scheme	Yes	Yes	Yes	Yes	Yes

Phase 2: $\mathcal{A}$ makes the second round of exposure queries and trapdoor queries as in phase 1, but the restriction is that the queried keyword cannot be mentioned in the challenge phase.

Guess: $\mathcal{A}$ runs the search to do a test. Finally, $\mathcal{A}$ sends a guess $b' \in \{0, 1\}$ to the challenger. $\mathcal{A}$ wins the game if $b = b'$.

Now, we analyze the probability that $\mathcal{C}$ can solve the (l, l)-BDHE problem. First, we define the following three events to simplify the probability analysis.

$E1$: $\mathcal{C}$ will not terminate during the exposure queries phase.

$E2$: $\mathcal{C}$ will not terminate during the trapdoor queries phase.

$E3$: $\mathcal{C}$ will not terminate during the challenge phase.

$E4$: $\mathcal{C}$ will not terminate during the simulation and it will produce the correct answer.

During the exposure queries phase, $\mathcal{C}$ will terminate if class $=$ helper. The probability that $\mathcal{A}$ chooses class $=$ helper is $1/2q_E$, then $\Pr[\text{class} \neq \text{helper}] = 1 - 1/2q_E \geq 1/q_E$. The probability that $\mathcal{C}$ will not terminate is at least $1/q_E$.

Meanwhile, from [20], we know that

$\Pr[E2] \geq 1/e$, $\Pr[E3] \geq 1/q_T$, $\Pr[E4] \geq \varepsilon/q_{H2}$

$$\begin{aligned} \varepsilon' &= \Pr[E1] \cap \Pr[E2] \cap \Pr[E3] \cap \Pr[E4] \\ &= \varepsilon/eq_E q_{H2} q_T. \end{aligned} \tag{21}$$

Since $\mathcal{C}$s success probability is at least $\varepsilon/eq_E q_{H2} q_T$, where e is base of natural logarithm.

VI. PERFORMANCE EVALUATION

We implement the scheme by using the MIRACL Core cryptography library in C++. The CS and devices are simulated on Ubuntu 18.04.3 with Intel Core i5-7500 CPU@3.40 GHz and 16 GB of memory. We choose a pairing-friendly elliptic curve BLS12-381 with embedding degree 12. Specifically, G_1 is the p-order subgroup of $E(F_p) : y^2 = x^3 + 4$ and G_2 is the p-order subgroup of $E'(F_{p^2}) : y^2 = x^3 + 4(u + 1)$ where the extension field F_{p^2} is defined as $F_p(u)/(u^2 + 1)$. And it can achieve 128-b security level.

A. Functional Comparison

In Table I, we compare the functions with several typical SE schemes. Mainly from these several aspects: MU, FS, BS, AC, and MK. From the Table I, we can see that our scheme realizes MU keyword SE, improves the tolerance to key exposure through key-insulated. In addition, our scheme combines AC and multi-key encryption (MK) to achieve secure and effective search and decryption.

TABLE II
RUNNING TIME(MS)

Users	Update	Encryt	Trapdoor	Search	Decrypt
10	1.085	8.805	3.978	15.601	11.315
12	1.085	8.833	3.972	15.59	11.33
14	1.097	8.818	3.981	15.635	11.3413
16	1.086	8.829	3.972	15.597	11.327
18	1.087	8.822	3.98	15.591	11.32
20	1.084	8.797	3.985	15.611	11.325
22	1.089	8.831	3.985	15.595	11.33
24	1.083	8.823	3.99	15.597	11.33
Average	1.0852	8.8197	3.9803	15.6021	11.3283

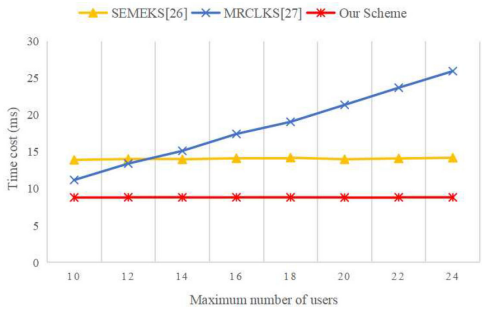


Fig. 3. Time cost of encrypt.

B. Computation Cost

In order to evaluate the performance of our scheme, we implement SEMEKS [26] and MRCLKS [27] in the same experimental environment. Because the comparison schemes lack some functions, we mainly compare the running time of encryption, trapdoor, search, and decryption. Table II shows the running times for our scheme with the different number of users. The average running time of the key update is 1.0852 ms.

Fig. 3 illustrates the time cost for the DO to run the encrypt. The encryption time of MRCLKS [27] is linearly related to the maximum number of users, while our scheme and SEMEKS [26] are not affected, which is approximately constant. When $n = 24$, the time cost of our scheme is about 8.823 ms, while that in SEMEKS [26] and MRCLKS [27] is 14.201 ms and 25.947 ms, respectively. By comparison, our scheme takes the least encryption time. Both our scheme and SEMEKS [26] utilize broadcast encryption to implement MUSE, while MRCLKS [27] uses the public key of all users to encrypt keywords, and the encryption overhead increases with the number of users. Therefore, for an IIoT system with numerous users but resource-constrained physical devices, our scheme has significant advantages.

Fig. 4 illustrates the time cost of the user to run the trapdoor. The time of generating the trapdoor of the three schemes is linearly related to the number of search keywords. When the number of keywords is 10, our scheme is about 39.9 ms, SEMEKS [26] and MRCLKS [27] is about 78.978 ms and 7.069 ms, respectively. In order to resist the harm of key exposure, our scheme takes more time to generate trapdoors than MRCLKS [27], but it is more effective than SEMEKS [26].

Then, we compare the time cost of the search at the CS-side. Fig. 5 illustrates the search time of MRCLKS [27] increases with

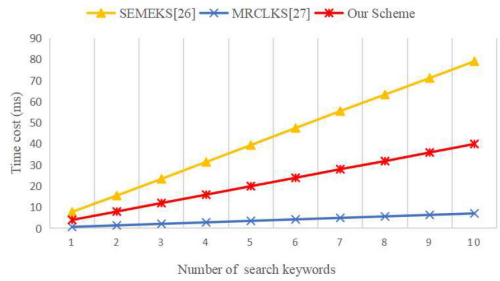


Fig. 4. Time cost of trapdoor.

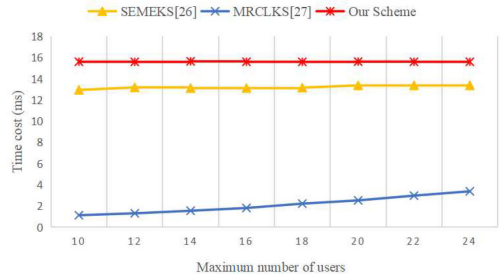


Fig. 5. Time cost of search.

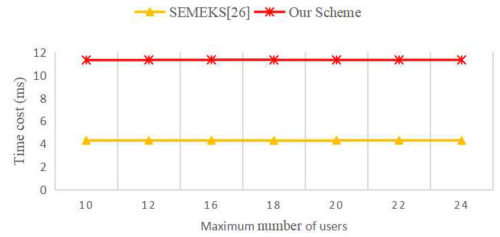


Fig. 6. Time cost of decrypt.

the number of users, while the search time of our scheme and SEMEKS [26] is approximately constant, and the time cost of our scheme is only a little more than the latter. When $n = 24$, the time is about 3.392 ms, 15.597 ms, and 13.371 ms, respectively. When the user key is updated, our scheme gives the permission of ciphertext update to the CS, which does not require the DO to re-encrypt the data online at all times. However, the cost is to increase the search time within an acceptable range.

Finally, from Fig. 6, we can see that the decryption time cost of our scheme and SEMEKS [26] is approximately constant. The main goal of this study is to improve the tolerance to key exposure by introducing a key-insulated primitive. Therefore, the decryption efficiency of our scheme is slightly lower than SEMEKS [26].

C. Communication Cost

Table III shows the contrast scheme communication cost, where $|G_1|$, $|G_2|$, $|G_T|$, $|Z_p^*|$, and h are the size of the elements in the elliptic curve group G_1 and G_2, an element in the bilinear target group G_T, an integer in Z_p^*, and a hash value, respectively. The sizes are 48, 192, 576, 48, and 20 B, respectively. During the encryption phase, for an increasing number of users, our scheme has greater advantages. Although it is slightly larger than the others for generating the trapdoor, our scheme provides a more secure search experience.

TABLE III
COMPARISON OF COMMUNICATION COST

Scheme	Encrypt size	Trapdoor size
SEMEKS [26]	$6\|G_1\|+2\|G_2\|+2\|G_T\|$	$\|G_1\|+4\|G_2\|$
MRCLKS [27]	$\|G_1\|+n\|Z_p^*\|+h$	$\|G_1\|+\|Z_p^*\|$
Our Scheme	$4\|G_1\|+2\|G_2\|+\|G_T\|$	$2\|G_1\|+4\|G_2\|$

VII. CONCLUSION

In this article, we proposed a PKI-MUSE scheme suitable for IIoT, which can solve the problem of ciphertext retrieval in an MU environment. We first introduce a key-insulated primitive in the MUSE and improve the tolerance to key exposure. Through experimental evaluation, our scheme was proven to have a higher computational efficiency and lower communication overhead during encryption. However, there is another limitation that the size of the master public key is considerably large, which linearly increases with the total number of users, and the total number of users is limited. These aspects need to be improved in the future.

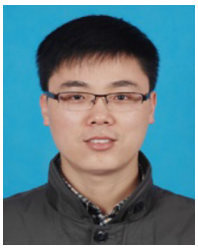
ACKNOWLEDGMENT

The authors would like to thank to the anonymous referees for their detailed comments and suggestions regarding this article.

REFERENCES

[1] Y. Liao, E. d. F. R. Loures, and F. Deschamps, "Industrial Internet of Things: A systematic literature review and insights," *IEEE Internet Things J.*, vol. 5, no. 6, pp. 4515–4525, Dec. 2018.
[2] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, 2015.
[3] J. Cui, F. Wang, Q. Zhang, Y. Xu, and H. Zhong, "An anonymous message authentication scheme for semi-trusted edge-enabled IIoT," *IEEE Trans. Ind. Electron.*, vol. 68, no. 12, pp. 12921–12929, Dec. 2021.
[4] M. Ma, D. He, M. K. Khan, and J. Chen, "Certificateless searchable public key encryption scheme for mobile healthcare system," *Comput. Elect. Eng.*, vol. 65, pp. 413–424, 2018.
[5] L. Wu, B. Chen, K.-K. R. Choo, and D. He, "Efficient and secure searchable encryption protocol for cloud-based Internet of Things," *J. Parallel Distrib. Comput.*, vol. 111, pp. 152–161, 2018.
[6] J. Cui, Y. Sun, Y. Xu, M. Tian, and H. Zhong, "Forward and backward secure searchable encryption with multi-keyword search and result verification," *Sci. China Inf. Sci.*, vol. 65, 2022, Art. no. 159102.
[7] H. Zhong, Z. Li, J. Cui, Y. Sun, and L. Liu, "Efficient dynamic multi-keyword fuzzy search over encrypted cloud data," *J. Netw. Comput. Appl.*, vol. 149, 2020, Art. no. 102469.
[8] H. Zhong, Z. Li, Y. Xu, Z. Chen, and J. Cui, "Two-stage index-based central keyword-ranked searches over encrypted cloud data," *Sci. China Inf. Sci.*, vol. 63, no. 3, pp. 1–3, 2020.
[9] Y. Wei, S. Lv, X. Guo, Z. Liu, Y. Huang, and B. Li, "FSSE: Forward secure searchable encryption with keyed-block chains," *Inf. Sci.*, vol. 500, pp. 113–126, 2019.
[10] Z. Zhang, J. Wang, Y. Wang, Y. Su, and X. Chen, "Towards efficient verifiable forward secure searchable symmetric encryption," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2019, pp. 304–321.
[11] X. Zhang, C. Xu, H. Wang, Y. Zhang, and S. Wang, "FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial Internet of Things," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 3, pp. 1019–1032, May/Jun. 2021.
[12] H. Kim, C. Hahn, and J. Hur, "Forward secure public key encryption with keyword search for cloud-assisted IoT," in *Proc. IEEE 13th Int. Conf. Cloud Comput.*, 2020, pp. 549–556.

[13] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Key-insulated public key cryptosystems," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2002, pp. 65–82.
[14] Y. Dodis, J. Katz, S. Xu, and M. Yung, "Strong key-insulated signature schemes," in *Proc. Int. Workshop Public Key Cryptogr.*, 2003, pp. 130–144.
[15] R. A. Popa and N. Zeldovich, "Multi-key searchable encryption," *IACR Cryptol. ePrint Arch.*, vol. 2013, pp. 508–525, 2013.
[16] Y. Miao, Q. Tong, R. Deng, K.-K. R. Choo, X. Liu, and H. Li, "Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2020.2989296.
[17] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surv.*, vol. 47, no. 2, pp. 1–51, 2014.
[18] Y. Wang, J. Wang, and X. Chen, "Secure searchable encryption: A survey," *J. Commun. Inf. Netw.*, vol. 1, no. 4, pp. 52–65, 2016.
[19] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.
[20] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 506–522.
[21] X. Tian and Y. Wang, "Id-based encryption with keyword search scheme from bilinear pairings," in *Proc. 4th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, 2008, pp. 1–4.
[22] 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 Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3618–3627, Aug. 2018.
[23] M. Ma, D. He, N. Kumar, K.-K. R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 759–767, Feb. 2018.
[24] N. Attrapadung, J. Furukawa, and H. Imai, "Forward-secure and searchable broadcast encryption with short ciphertexts and private keys," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2006, pp. 161–177.
[25] M. Ali, H. Ali, T. Zhong, F. Li, Z. Qin, and A. A. Ahmed Abdelrahaman, "Broadcast searchable keyword encryption," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, 2014, pp. 1010–1016.
[26] A. Kiayias, O. Oksuz, A. Russell, Q. Tang, and B. Wang, "Efficient encrypted keyword search for multi-user data sharing," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2016, pp. 173–195.
[27] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multirecipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2553–2562, Apr. 2020.
[28] G. Hanaoka, Y. Hanaoka, and H. Imai, "Parallel key-insulated public key encryption," in *Proc. Int. Workshop Public Key Cryptogr.*, 2006, pp. 105–122.
[29] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. 1st ACM Conf. Comput. Commun. Secur.*, 1993, pp. 62–73.
[30] S. Patranabis, Y. Shrivastava, and D. Mukhopadhyay, "Dynamic key-aggregate cryptosystem on elliptic curves for online data sharing," in *Proc. Int. Conf. Cryptol. India*, 2015, pp. 25–44.



Jie Cui was born in Henan Province, China, in 1980. He received the Ph.D. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2012.

He is currently a Professor and Ph.D. supervisor with the School of Computer Science and Technology, Anhui University, Hefei. He has authored or coauthored more than 120 scientific publications in reputable journals (e.g., IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON MULTIMEDIA), academic books, and international conferences. His research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.

Jie Lu received the B.Eng. degree in computer science and technology from Anhui Normal University Wanjiang College, Wuhu, China, in 2019.

She is currently a Research Student with the School of Computer Science and Technology, Anhui University, Hefei, China. Her research interests include the security of industrial Internet of Things.

Hong Zhong was born in Anhui Province, China, in 1965. She received the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2005.

She is currently a Professor and Ph.D. supervisor with the School of Computer Science and Technology, Anhui University, Hefei. She has authored or coauthored more than 150 scientific publications in reputable journals (e.g., IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON MULTIMEDIA, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON CLOUD COMPUTING, and IEEE TRANSACTIONS ON BIG DATA), academic books, and international conferences. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security, and software-defined networking.

Qingyang Zhang was born in Anhui Province, China, in 1992. He received the Ph.D. degree in computer science and technology from Anhui University, Hefei, China, in 2021.

He is currently a Lecturer with the School of Computer Science and Technology, Anhui University. His research interests include edge computing and computer systems and security.

Chengjie Gu received the Ph.D. degree in information and communication engineering from the Nanjing University of Posts and Telecommunications, Nanjing, China, in 2012.

From 2012 to 2017, he was an Innovation Team Leader with the 38th Research Institute of CETC and conducted research and development in the communication and networking sector. He is currently a President with Security Research Institute, New H3C Group, Hefei, China. He is also studying for Postdoctoral Fellowship with the University of Science and Technology of China, Hefei. He is a high-level Innovation Leader of Anhui Province and a cybersecurity expert of Zhejiang Province in China. His research interests include network security and trusted network architecture.

Lu Liu received the M.Sc. degree in data communication systems from Brunel University, Uxbridge, U.K., in 2003, and the Ph.D. degree in space center computers from the University of Surrey, Guildford, U.K., in 2007.

He is currently a Professor of informatics and the Head with the School of Informatics, University of Leicester, Leicester, U.K. His research interests include areas of cloud computing, service computing, computer networks, and peer-to-peer networking.

Dr. Liu is a Fellow of British Computer Society.