IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 6 May 2023, accepted 4 June 2023, date of publication 9 June 2023, date of current version 16 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3285114

RESEARCH ARTICLE

A Pairing-Free Certificateless Searchable Public Key Encryption Scheme for Industrial Internet of Things

XIAOGUANG LIU[1,2,3], HAO DONG[2,3], NEHA KUMARI[4], AND JAYAPRAKASH KAR[4], (Senior Member, IEEE)

[1]School of Mathematics, Southwest Minzu University, Chengdu, Sichuan 610041, China
[2]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, Guangxi 541004, China
[3]Key Laboratory for Computer Systems of State Ethnic Affairs Commission, Southwest Minzu University, Chengdu, Sichuan 610041, China
[4]Centre for Cryptography, Cyber Security and Digital Forensics, Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur, Rajasthan 302031, India

Corresponding author: Xiaoguang Liu (21700128@swun.edu.cn)

This work was supported in part by the Sichuan Science and Technology Program under Grant 2023NSFSC0471, in part by the Fund of the Guangxi Key Laboratory of Cryptography and Information Security under Grant GCIS202121, in part by the Foreign Experts Program of Ministry of Science and Technology of China under Grant DL2022186001L, and in part by the Fundamental Research Funds for the Central Universities of Southwest Minzu University under Grant ZYN2022077.

ABSTRACT The Industrial Internet of Things (IIoT) collects a large amount of data through various types of sensors and intelligently processes this data using cloud computing, which is flexible, efficient, and cost-effective. Since IIoT data is stored on the cloud service provider's server, the data must be encrypted to protect the user's privacy. However, the encrypted data faces the search problem, which is usually solved by Public Key Encryption with Keyword Search (PEKS). In addition, most existing PEKS schemes are vulnerable to Inside Keyword Guessing Attacks (IKGA). Recently, some certificateless public key authenticated encryption with keyword search (CLPEKS) schemes have been proposed, which not only avoid the problems of certificate management and key escrow but can also resist IKGA. However, most of them rely on the expensive bilinear pairing. To overcome these problems, in this paper we propose a pairing-free CLPEKS scheme. The security of the proposed scheme is proved in the random oracle model. The analysis results show that the proposed scheme has better overall performance in terms of computational cost, communication cost and security properties.

INDEX TERMS Pairing-free, certificateless, searchable public key encryption, security, IIoT.

I. INTRODUCTION

The Internet of Things (IoT) is the integration of sensors, software, and smart devices [1]. It connects things with the Internet to realize intelligent identification, positioning, tracking, and management by sharing environment and status information in real-time. Nowadays, more applications of IoT are in the industrial sector, and it is the key to the realization of intelligent manufacturing [2]. In the manufacturing industry, IIoT plays an important role in quality control, supply chain traceability, and overall supply chain efficiency. However, storing all raw data on local IIoT devices is not suitable due to the severely limited energy consumption and storage space of the end devices. Fortunately, cloud servers can provide flexible data storage. IIoT collects large amounts of fragmented data through sensors, and cloud computing can store and intelligently process this data. This eliminates the need for users to maintain expensive hardware and dedicated space. Fig. 1 shows a typical network architecture for cloud-assisted IIoT.

In recent years, the combination of IIoT and cloud computing has developed rapidly [3], [4], but because cloud computing requires users to outsource data to cloud service providers, users have only partial control over this data [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Jun Wang.

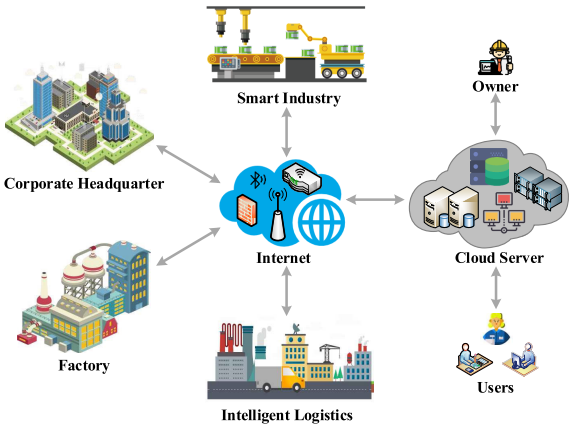

FIGURE 1. Typical network architecture for cloud-assisted IIoT.

The cloud service provider may extract and manipulate the users' data because the provider is not fully trustworthy. In general, to solve the security of outsourced data, IIoT data should be encrypted before being uploaded to the cloud server. As such, encryption technology plays a critical role in protecting the privacy and data integrity of industrial enterprises. However, common encryption methods have not addressed the issue of searching for encrypted data. One undesirable method is for users to download all the encrypted data and then decrypt the information needed to search. The computational and communication costs increase exponentially with the amount of encrypted data, which is not feasible in practical applications. In 2000, Song et al. [6] first proposed Searchable Encryption (SE), which was the first symmetric SE. It allows authorized users to hide queries without revealing any keywords to an untrusted cloud server. A vexing problem with it is how to secretly distribute keys to authorized users [7]. In addition, this interactive scheme inevitably requires higher computational and communication cost. To solve these problems, Boneh et al. [8] proposed a non-interactive scheme, i.e., the first PEKS scheme. It uses the data user's public key to generate a trapdoor containing the ciphertext of the keywords search index and sends it to the cloud server. The data user then uses his/her private key to generate a corresponding search query ciphertext and sends it to the cloud server. Only if the trapdoor matches the keyword in the query ciphertext will the cloud server return the corresponding ciphertext to the data user.

Unfortunately, many existing PEKS schemes have been proven to be unreliable because they are vulnerable to IKGA [9], [10]. The cloud server can recover keywords by guessing common keywords offline, as the keyword space is usually limited. To solve this problem, Huang and Li [11] proposed public key authenticated encryption with keyword search. It adds authentication to the data user compared to general PEKS schemes. Therefore, if the cloud server is not authorized and it is difficult to start IKGA. However, most schemes are based on traditional Public Key Infrastructure (PKI) or Identity Cryptography (IBC) [12]. PKI is not only expensive to spend a lot of cost to manage certificates, but also vulnerable to denial of service attacks. There are several key escrow problems in IBC. Therefore, He et al. [13] and Lu and Li [14] respectively proposed a CLPEKS scheme, which not only solves the problems of certificate management and key escrow but can also resist IKGA. However, they rely on expensive bilinear pairing. Recently, Danial et al. [15] proposed a pairing-free certificateless authenticated encryption with keyword search scheme. This scheme is only secure if the two KGC do not cooperate.

In this paper, we propose a pairing-free CLPEKS scheme for IIoT. This scheme does not have the certificate management problem of PKI or the key escrow problem as it is based on certificateless public key cryptography (CLPKC). Notably, the proposed scheme not only does not use costly bilinear pairing but also can resist IKGA. The main research contributions are as follows.

- We propose a CLPEKS scheme that avoids bilinear pairing with high computational cost. This is important because IIoT devices are often resource-constrained.
- The proposed scheme can resist IKGA. The reason is that it adds salt to the trapdoor, and the search frequency of the keywords is not exposed to the cloud server. In addition, this scheme allows the data owner to encrypt the keywords and authenticate the identity of the data user. Therefore, the cloud server that is not authorized cannot start IKGA.
- The security of the proposed scheme is proved in the random oracle model. The analysis results show that our scheme is more secure, and has better overall performance, although it doesn't have the advantage in communication cost.

II. RELATED WORKS

To address the problem of searching encrypted data, Boneh et al. [8] designed the first PEKS scheme, which is constructed based on bilinear pairing. Since then, many improved PEKS schemes have been proposed [9], [16], [17]. However, Byun et al. [18] showed that many PEKS schemes are vulnerable to offline keyword guessing attacks (OKGA). This is because users typically use common keywords when searching for documents, and the min-entropy of these keywords is too low. Therefore, Rhee et al. [19] proposed a PEKS scheme with a designated tester who restricts the user's query conditions to prevent trapdoors from leaking too much keyword information. Wang et al. [20] pointed out that [19] cannot resist OKGA from malicious servers. Tang et al. [21] proposed the public key encryption with registered keyword search, which requires the data owner to register a keyword and send it to the user in advance. Although it was shown to resist OKGA, keyword pre-registration may introduce other problems. Xu et al. [22] designed the public key encryption with fuzzy keyword search, where multiple keywords share a fuzzy keyword trapdoor. However, Liu et al. [23] pointed out that these schemes can only resist OKGA initiated by external adversaries, but cannot resist IKGA initiated by internal adversaries (such as cloud service providers). To address

this issue, Chen et al. [24] presented a dual-server PEKS scheme, which requires that the two cloud servers do not collude. Recently, Lu et al. [25] presented a new certificate-based searchable encryption scheme. Abdalla et al. [26] combined IBC and PEKS to propose identity-based encryption with keyword search. Later, Li et al. [27] and Qin et al. [28] respectively presented identity-based PEKS schemes authorized to users. However, IBC has a number of key escrow problems because the user's private key is only generated by the private key generator (PKG).

In 2003, Al-Riyami and Paterson [29] proposed CLPKC. In CLPKC, the key generation center (KGC) uses the user's identity to generate a partial private key. The user computes a full private key by combining the partial private key with a secret value of his/her own choice, so that the KGC does not know the full private key. Therefore, CLPKC not only solves the key escrow problem in IBC but also does not require any digital certificates. In 2014, Peng et al. [30] first designed a CLPEKS scheme. In 2017, He et al. [13] designed a CLPEKS scheme for IIoT. In 2018, Ma et al. [31] also gave a CLPEKS scheme for IIoT, but Zhang et al. [32] found that the scheme in [31] cannot satisfy the trapdoor indistinguishability. In addition, many pairing-free certificateless CLPEKS schemes have been proposed, which are more suitable for IIoT applications. In 2019, Lu et al. [33] constructed a pairing-free certificateless CLPEKS scheme. However, Ma et al. [34] found that the scheme in [33] is vulnerable to user impersonation attacks and proposed a new pairing-free dual-server CLPEKS scheme for cloud-based IIoT. In 2020, Lu et al. [35] found that the keyword is encrypted by the receiver's public key in [31], and the adversary can impersonate the sender to encrypt the keyword and initiate IKGA. They proposed a privacy-preserving and pairing-free multirecipient CPEKS scheme. However, the malicious server can easily compute the user's private key by accessing the trapdoor. Recently, [36] and [37] respectively presented a secure and efficient pairing-free CLPEKS scheme.

Collectively, the existing PEKS schemes typically have the following limitations: (1) Many schemes have a certificate management problem or a key escrow problem. (2) Most of the schemes cannot resist keyword guessing attacks from internal or external adversaries. (3) Many schemes are based on expensive bilinear pairing and cannot be practically applied to lightweight devices.

In this paper, we propose an ECC-based CLPEKS scheme. Since it is pairing-free, it is lightweight and efficient enough to be more easily implemented in the IIoT. The scheme not only satisfies various security performances but also makes it easy for users to search for encrypted data.

III. PRELIMINARIES

A. ELLIPTIC CURVE CRYPTOGRAPHY

The elliptic curve E is a plane curve defined on the prime finite field $\mathbb{F}_p$, which is generally defined by the Weierstrass equation

$$y^2 = x^3 + ax + b \tag{1}$$

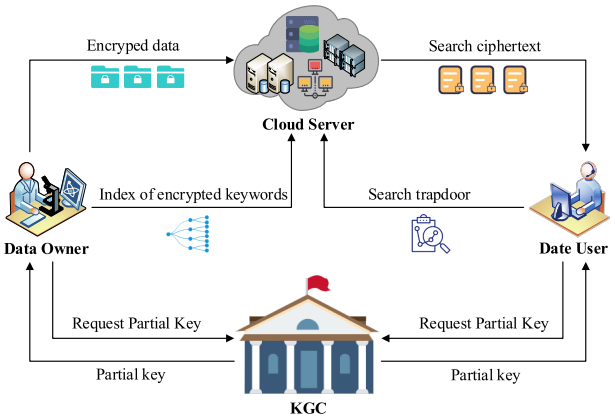

FIGURE 2. Basic system model for the proposed CLPEKS scheme.

with $a, b \in \mathbb{F}_p$ and the discriminant $\Delta = \left(4a^3 + 27b^2\right) \bmod p \neq 0$. The space of points is denoted by $E\left(\mathbb{F}_p\right)$, which together with an infinite point O form a group

$$\mathbb{G} = \left\{(x, y) : x, y \in \mathbb{F}_p, (x, y) \in E\left(\mathbb{F}_p\right)\right\} \cup \{O\}. \tag{2}$$

$\mathbb{G}$ is a cyclic additive group under the point addition "+" defined as follows: let $P, Q \in \mathbb{G}$, and L is the straight line to $E\left(\mathbb{F}_p\right)$ connecting P and Q. If $P = Q$, L is the tangent line to $E\left(\mathbb{F}_p\right)$. Let $P + Q = R$, where R is the third point of intersection of the line l with $E\left(\mathbb{F}_p\right)$. If there is no third intersection of L with $E\left(\mathbb{F}_p\right)$, then $P + Q = O$. The scalar multiplication over $E\left(\mathbb{F}_p\right)$ is computed as follows:

$$tP = P + P + \cdots + P_{(t\ times)}. \tag{3}$$

B. HARDNESS ASSUMPTION

The security of the proposed scheme is based on the hardness assumption of the computational Diffie-Hellman (CDH) problem.

Definition 1 (CDH): Let $\mathbb{G}$ be an elliptic curve group of prime order q. Given the generator P of $\mathbb{G}$ and (aP, bP), it is difficult to compute abP, where $a, b \in \mathbb{Z}_q^*$ are unknown numbers.

The hardness assumption of the CDH problem is that the advantage $adv^{CDH}(A)$ can be negligible for any polynomial time algorithm A, where $adv^{CDH}(A) = Pr[A(p, aP, bP) = abP | a, b \in \mathbb{Z}_q^*]$.

C. SYSTEM MODEL

As shown in Fig. 2, the system model of the CLPEKS scheme consists of four entities, namely the KGC, the cloud server, the data owner, and the data user. Each entity operates as follows:

1) *KGC*: It is a trusted entity that generates the public system parameters and computes the partial private key by obtaining the identity information of the data owner/data user.
2) *Data Owner*: It first extracts the keywords from the encrypted data and creates an index. It then encrypts the keywords index using its private key and the data user's public key. Finally, it stores the encrypted data and the keywords index on the cloud server.

3) *Data User*: It uses its private key and the data owner's public key to generate a trapdoor for the keywords. It then sends the trapdoor to the cloud server. Finally, it uses its key to decrypt the returned ciphertext.
4) *Cloud Server*: An honest-but-curious entity that honestly implements a predefined protocol but is curious about the stored data information. It is responsible for testing the trapdoor and keywords index according to predefined protocols.

The workflow of the system model is given as follows.

1) *Setup*: KGC initializes public parameters and master keys according to security parameters.
2) *Key generation*: After receiving the request from the data owner or the data user, KGC uses the master key to generate part of the private key and sends it to the data owner or the data user. After receiving part of the private key, the data owner or the data user selects a secret value as another part of the private key and synthesizes the complete private key and generates the public key.
3) *Encryption*: The data owner first encrypts the data file, extracts the keywords from the data file and builds the keywords index. Finally, the ciphertext and the keywords index are sent to the cloud server.
4) *Trapdoor generation*: The data user generates the trapdoor based on the queried keywords and sends it to the cloud server.
5) *Test*: The cloud server uses the uploaded trapdoor to search the keywords index. If no keyword matches, it returns false. Otherwise, the matching ciphertext is sent to the data user.

D. FORMAL DEFINITION

The proposed CLPEKS scheme consists of the following eight probabilistic polynomial-time algorithms.

1) *Setup* (λ): It takes a security parameter λ as input, and outputs a system master key s and a set of global system parameters *params*. The algorithm is run by KGC.
2) *Partial-Key-Extract* (*params*, s, ID_i): It takes *params*, s, and the user's ID_i as input, and returns a partial public/private key (R_{ID_i}, d_{ID_i}). The algorithm is performed by KGC. Here, ID_i may be is the identity ID_o of the data owner or the identity ID_u of the data user.
3) *Set-Secret-Value* (*params*, ID_i): It takes *params* and a user's ID_i as input and returns a secret value x_{ID_i}. The algorithm is executed by the user.
4) *Set-Private-Key* (*params*, ID_i, d_{ID_i}): It takes *params*, a user's ID_i, and d_{ID_i} as input, and outputs a complete private key SK_{ID_i}. The algorithm is run by the user.
5) *Set-Public-Key* (*params*, R_{ID_i}, ID_i, x_{ID_i}): It takes *params*, a partial public R_{ID_i}, a user's ID_i, and x_{ID_i} as input, and outputs a public key PK_{ID_i}. The algorithm is run by the user.
6) *Encrypt* (*params*, w_i, SK_{ID_o}, PK_{ID_u}): It takes *params*, the keyword w_i, SK_{ID_o}, and PK_{ID_u} as input, and outputs the ciphertext C of the keyword w_i. The algorithm is executed by the data owner.
7) *Trapdoor* (*params*, w, PK_{ID_o}, SK_{ID_u}): It takes *params*, a search keyword w, PK_{ID_o}, and SK_{ID_u} as input and returns the trapdoor T_w. The algorithm is executed by the data user.
8) *Test* (*params*, T_w, C, PK_{ID_o}, PK_{ID_u}): It takes *params*, the trapdoor T_w, the ciphertext C, PK_{ID_o}, and PK_{ID_u} as input, and outputs 1 if C and T_w contain the same keyword. Otherwise, it returns 0. The algorithm is executed by the cloud server.

E. SECURITY MODEL

The CLPEKS scheme is semantically secure for indistinguishability against chosen keyword attacks (hereafter referred to as IND-CLPEKS-CKA), defined as follows.

In CLPKC, the adversary can query the partial and full private keys of the chosen identity. However, there is no public key certificate, so the adversary can replace the user's public key. Therefore, CLPKC divides adversaries into Type I adversary $\mathcal{A}_I$ and Type II adversary $\mathcal{A}_{II}$. $\mathcal{A}_I$ simulates an honest but curious cloud server or malicious user, who does not know the master key s, but it can replace the user's public key at will. $\mathcal{A}_{II}$ simulates a malicious KGC that knows the master key s, but it cannot replace the user's public key. According to two different adversaries, $\mathcal{A}_I$ and $\mathcal{A}_{II}$ in CLPKC, the game between the challenger $\mathcal{C}$ and the adversary is also divided into two types to define the security model of the proposed scheme, which are described separately below.

Game 1: Game 1 is the interaction between $\mathcal{C}$ and $\mathcal{A}_I$.

1) *Setup*: This takes a security parameter λ as input, and $\mathcal{C}$ sends the public system parameter *params* to $\mathcal{A}_I$. $\mathcal{C}$ keeps the master key s secret.
2) *Hash-Query*: $\mathcal{A}_I$ is allowed to adaptively query the oracle and gets the hash value queried.
3) *Partial-Key-Extract-Query*: $\mathcal{A}_I$ selects an identity ID_i as input, $\mathcal{C}$ runs the Partial-Key-Extract algorithm and returns the partial public/private key (R_{ID_i}, d_{ID_i}) corresponding to identity ID_i to $\mathcal{A}_I$.
4) *Private-Key-Extract-Query*: $\mathcal{A}_I$ chooses an identity ID_i as input. If the public key of the identity ID_i has not been replaced, $\mathcal{C}$ runs the Set-Private-Key algorithm and returns the complete private key corresponding to ID_i to $\mathcal{A}_I$.
5) *Request-Public-Key-Query*: $\mathcal{A}_I$ selects an identity ID_i as input, $\mathcal{C}$ runs the Set-Public-Key algorithm and sends the public key PK_{ID_i} corresponding to the identity ID_i to $\mathcal{A}_I$.
6) *Replace-Public-Key-Query*: $\mathcal{A}_I$ is allowed to use the public key PK'_{ID_i} of its choice to replace the original public key PK_{ID_i} of the identity ID_i.
7) *Trapdoor-Query*: $\mathcal{A}_I$ submits a keyword w for query, $\mathcal{C}$ runs the Trapdoor algorithm and sends the corresponding the trapdoor T_w to $\mathcal{A}_I$.
8) *Challenge*: $\mathcal{A}_I$ generates two challenge keywords (w_0, w_1) and an identity ID^* that wants to be challenged. The identity ID^* is not allowed to be an identity

TABLE 1. Notations.

Notations	Description	Notations	Description
q	large prime order	$\mathbb{G}$	the cyclic additive group
p	large prime order	P	Generator of $\mathbb{G}$
λ	security parameter	$r_{(.)}$	random number
s	system master key	$d_{(.)}$	partial private key
$ID_{(.)}$	identity	$R_{(.)}$	partial public key
$PK_{(.)}$	public key	$H_{(.)}$	hash function to point in $\mathbb{Z}_q^*$
$SK_{(.)}$	private key	x_1	the x-coordinate of point kP
$\mathbb{F}_p$	the prime finite field	y_1	the y-coordinate of point kP

that has executed Private-Key-Extract-Query, nor is it allowed to be an identity that has executed Replace-Public-Key-Query and Partial-Key-Extract-Query. The challenger $\mathcal{C}$ randomly selects $\gamma \in \{0, 1\}$, executes the Encrypt algorithm to generate ciphertext C^*, and sends it to $\mathcal{A}_I$. If the ciphertext cannot be generated, then $\mathcal{A}_I$ fails in this game.

9) *Guess*: $\mathcal{A}_I$ enters its guess, which is a bit $\gamma' \in \{0, 1\}$. If $\gamma' = \gamma$, then $\mathcal{A}_I$ wins the game. The advantage of $\mathcal{A}_I$ is defined as $Adv(\mathcal{A}_I) = \left|\Pr\left[\gamma' = \gamma\right] - 1/2\right|$, where $\Pr\left[\gamma' = \gamma\right]$ is the probability of $\gamma' = \gamma$.

Game 2: Game 2 is the interaction between $\mathcal{C}$ and $\mathcal{A}_{II}$.

1) *Setup*: It takes a security parameter λ as input, $\mathcal{C}$ sends the generated system parameter *params* and the master key s to $\mathcal{A}_{II}$.
2) *Hash-Query*: These are the same queries as in game 1.
3) *Partial-Key-Extract-Query*: These are the same queries as in game 1.
4) *Trapdoor-Query*: These are the same queries as in game 1.
5) *Challenge*: $\mathcal{A}_{II}$ generates two challenge keywords (w_0, w_1) and an identity ID^* that wants to be challenged. The identity ID^* is not allowed to be an identity that has executed Partial-Key-Extract-Query. The challenger $\mathcal{C}$ randomly selects $\gamma \in \{0, 1\}$, executes the Encrypt algorithm to generate ciphertext C^*, and sends it to $\mathcal{A}_{II}$.
6) *Guess*: $\mathcal{A}_{II}$ inputs its guess, which is a bit $\gamma' \in \{0, 1\}$. $\mathcal{A}_{II}$ wins the game if $\gamma' = \gamma$. The advantage of $\mathcal{A}_{II}$ is defined as $Adv(\mathcal{A}_{II}) = \left|\Pr\left[\gamma' = \gamma\right] - 1/2\right|$.

Definition 2: The CLPEKS scheme is IND-CLPEKS-CKA secure if the advantages $Adv(\mathcal{A}_I)$ and $Adv(\mathcal{A}_{II})$ can be omitted.

IV. THE PROPOSED CLPEKS SCHEME

In this section, we will give a CLPEKS scheme based on CLPKC. The scheme does not need the expensive bilinear pairing. The notations used in this paper are given in Table 1. The proposed CLPEKS scheme consists of eight polynomial-time algorithms as follows. We denote the data owner's identity as ID_o and the data user's identity as ID_u.

1) **Setup**(λ): Given a security parameter λ, KGC selects a cyclic group $\mathbb{G}$ of prime order q on the elliptic curve $E\left(\mathbb{F}_p\right)$, where p is the scale of the finite field F_p. Let the base point P be the generator of $\mathbb{G}$. KGC chooses a random number $s \in \mathbb{Z}_q^*$ as the master key, and sets the master public key $P_{pub} = sP$. Then, KGC selects three secure hash functions H_1:$\{0, 1\}^* \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$, H_2:$\{0, 1\}^* \rightarrow \mathbb{Z}_q^*$, and H_3:$\{0, 1\}^* \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{Z}_q^*$. Finally, KGC releases the system parameters $params = \left\{E\left(\mathbb{F}_q\right), p, q, \mathbb{G}, P, P_{pub}, H_1, H_2, H_3\right\}$ and keeps s secret.
2) **Partial-Key-Extract**($params$, s, ID_i): Taking an entity's identity $ID_i \in \{0, 1\}^*$ (ID_i may be ID_o or ID_u) as input, KGC selects a random number $r_{ID_i} \in \mathbb{Z}_q^*$, computes $R_{ID_i} = r_{ID_i}P$, $\alpha_{ID_i} = H_1\left(ID_i, R_{ID_i}\right)$, and extracts $d_{ID_i} = r_{ID_i} + s\alpha_{ID_i} \pmod p$ as the partial private key. KGC returns a partial public/private key (R_{ID_i}, d_{ID_i}) to the entity. The entity can be the data owner or the data user.
3) **Set-Secret-Value**($params, ID_i$): The entity generates a random number $x_{ID_i} \in \mathbb{Z}_q^*$ as its secret value.
4) **Set-Private-Key**($params, ID_i, d_{ID_i}$): Taking the partial private key d_{ID_i} and the secret value x_{ID_i} as input, the entity generates $SK_{ID_i} = \left(x_{ID_i}, d_{ID_i}\right)$ as its complete private key.
5) **Set-Public-key**($params$, R_{ID_i}, ID_i, x_{ID_i}): Taking the partial public key R_{ID_i} as input, the entity computes $P_{ID_i} = x_{ID_i}P$, and generates its full public key $PK_{ID_i} = \left(P_{ID_i}, R_{ID_i}\right)$.
6) **Encrypt**($params$, w_i, SK_{ID_o}, PK_{ID_u}): Let a set of keywords $W = \{w_i | 1 \le i \le n\}$. Taking $params, SK_{ID_o}$, and PK_{ID_u} as input, the data owner encrypts the keyword $w_i \in W$ as shown below:
   a) Generates a random number $k \in \mathbb{Z}_q^*$ and computes $kP = (x_1, y_1)$, $c_1 = x_1 \bmod p$. If $c_1 = 0$, the data owner chooses another random number k and re-executes the above process.
   b) Computes the authorization tokens

$$2\beta_{ou} = H_3\left(au_{ou}, ID_o, P_{ID_o}, R_{ID_o}, ID_u, P_{ID_u}, R_{ID_u}\right), \quad (4)$$

$$c_2 = \beta_{ou}\left(P_{ID_u} + R_{ID_u} + \alpha_{ID_u}P_{pub}\right), \quad (5)$$

   where

$$au_{ou} = \left(x_{ID_o} + d_{ID_o}\right)\left(P_{ID_u} + R_{ID_u} + \alpha_{ID_u}P_{pub}\right) = \left(x_{ID_o} + d_{ID_o}\right)\left(x_{ID_u} + d_{ID_u}\right)P. \quad (6)$$

   c) Computes

$$c_3 = k^{-1}\left(H_2\left(w_i\right) + \beta_{ou}c_1\left(x_{ID_o} + d_{ID_o}\right)\right) \bmod p. \quad (7)$$

   If $c_3 = 0$, the data owner chooses another random number k and re-executes the above process.

Finally, the data owner sends the ciphertext $C = (c_1, c_2, c_3)$ to the cloud server.

7) **Trapdoor**($params$, w, PK_{ID_o}, SK_{ID_u}): Taking $params$, a search keyword w, PK_{ID_o}, and SK_{ID_u} as input, the data user computes the trapdoor T_w as follows:
   a) Generates a random number $r_u \in \mathbb{Z}_q^*$, and sets $t_1 = r_u$. If $r_u = 0$, the data user selects another random number r_u.
   b) Computes the authentication tokens

$$2\beta_{uo} = H_3\left(au_{uo}, ID_o, P_{ID_o}, R_{ID_o}, ID_u, P_{ID_u}, R_{ID_u}\right), \quad (8)$$

$$t_2 = \beta_{uo}\left(P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub}\right), \quad (9)$$

   where

$$\begin{aligned} 2au_{uo} &= \left(x_{ID_u} + d_{ID_u}\right)\left(P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub}\right) \\ &= \left(x_{ID_u} + d_{ID_u}\right)\left(x_{ID_o} + d_{ID_o}\right) P. \end{aligned} \quad (10)$$

   c) Computes

$$\begin{aligned} 2t_3 = \left(1 + \beta_{uo}\left(x_{ID_u} + d_{ID_u}\right)\right)^{-1} &\left(H_2\left(w-\right)\right. \\ &\left. t_1\left(x_{ID_u} + d_{ID_u}\right)\right) \bmod p. \end{aligned} \quad (11)$$

   If $t_3 = 0$, the data user selects another random number r_u and re-executes the above process. Finally, the data user sends $T_w = (t_1, t_2, t_3)$ to the cloud server.

8) **Test**($params$, T_w, C, PK_{ID_o}, PK_{ID_u}): Taking $params$, T_w, C, PK_{ID_o}, and PK_{ID_u} as the input, the cloud server will compute the elliptic curve point $\left(x_1', y_1'\right) = c_3^{-1}\left(t_3 P + t_1\left(P_{ID_u} + R_{ID_u} + \alpha_{ID_u} P_{pub}\right) + t_3 c_2\right) + c_1 c_3^{-1} t_2$ and checks whether $c_1 = x_1'$ is true. If the equation is true, it outputs "1". Otherwise, it outputs "0".

Correctness: Let T_w be the trapdoor of the keyword w, and w_i be the keyword contained in the ciphertext C. If $w = w_i (i \in \{1, 2, \cdots, n\})$, then the equation (12), as shown at the bottom of the next page, is true. So, we can see that only if $H_2(w) = H_2(w_i)$, the correct elliptic curve point $\left(x_1', y_1'\right)$ can be computed.

V. SECURITY ANALYSIS

In this section, we will prove that the proposed CLPEKS scheme is provably secure in the random oracle model. Theorem 1 can be derived from Lemma 1 and Lemma 2.

Theorem 1: In the random oracle model, when the CDH problem is hard, the proposed CLPEKS scheme is semantically secure for IND-CLPEKS-CKA.

Lemma 1: Suppose that the adversary $\mathcal{A}_I$ breaks the non-interactive CLPEKS scheme for IND-CLPEKS-CKA with a non-negligible advantage ε. $\mathcal{A}_I$ performs at most q_{H_i} $(i = 1, 2, 3)$, q_{pp}, q_{ke}, and q_t times H_i $(i = 1, 2, 3)$-Query, Partial-Key-Extract-Query, Private-Key-Extract-Query, and Trapdoor-Query, respectively. We use the ability of $\mathcal{A}_I$ to construct a polynomial time-bounded algorithm $\mathcal{C}$ to solve the CDH problem with an advantage

$$\varepsilon' \geq \frac{\varepsilon}{q_{H_1} q_{H_2} q_{H_3}}\left(1 - \frac{1}{q_{H_1}}\right)^{q_{pp}+q_{ke}+q_t}. \quad (13)$$

Proof: Suppose that in the polynomial-time T, the challenger $\mathcal{C}$ first gives the generator P of a group $\mathbb{G}$. Next, $\mathcal{C}$ chooses two random numbers $a, b \in \mathbb{Z}_q^*$. Finally, $\mathcal{C}$ is given an instance (aP, bP) to compute the CDH problem, and constructs a simulator $\mathcal{S}$ ($\mathcal{S}$ acts as the challenger $\mathcal{C}$ in the game) to compute $cP = abP \bmod n$. ■

1) *Setup*: $\mathcal{S}$ runs Setup, sets the system public key $P_{pub} = aP$, but cannot compute the value of a. In addition, $\mathcal{S}$ randomly chooses ID_I $\left(1 \leq I \leq q_{H_1}\right)$ as the challenger identity. Finally, $\mathcal{S}$ generates system parameters $params = \left\{E\left(\mathbb{F}_q\right), q, \mathbb{G}, P, P_{pub}, H_1, H_2, H_3\right\}$, and then sends $params$ to the adversary $\mathcal{A}_I$. $\mathcal{S}$ responds to the query from $\mathcal{A}_I$ as follows.
2) H_1-*Query*: $\mathcal{S}$ holds a list called L_{H_1}, which contains a tuple of the form $\langle ID_i, \alpha_i, R_{ID_i}\rangle$. $\mathcal{A}_I$ queries $H_1(ID_i)$, if identity ID_i already exists in the list L_{H_1} combination, $\mathcal{S}$ outputs α_i to $\mathcal{A}_I$. Otherwise, $\mathcal{S}$ selects a random number $\alpha_i \in \mathbb{Z}_q^*$, and adds the tuple $\langle ID_i, \alpha_i, R_{ID_i}\rangle$ to the list L_{H_1}, then returns $H_1\left(ID_i, R_{ID_i}\right) = \alpha_i$ to $\mathcal{A}_I$.
3) H_2-*Query*: $\mathcal{S}$ holds a list L_{H_2} containing a tuple of the form $\langle w_i, H_2(w_i), h_{i2}\rangle$. $\mathcal{A}_I$ queries $H_2(w_i)$, if the identity ID_i already exists in the list L_{H_2} combination, $\mathcal{S}$ outputs h_{i2} to $\mathcal{A}_I$. Otherwise, $\mathcal{S}$ chooses a random number $h_{i2} \in \mathbb{Z}_q^*$, and adds the tuple $\langle w_i, H_2(w), h_{i2}\rangle$ to the list L_{H_2}, then returns $H_2(w_i) = h_{i2}$ to $\mathcal{A}_I$.
4) H_3-*Query*: $\mathcal{S}$ holds a list called L_{H_3} containing a tuple of the form $\langle au_i, ID_i, P_{ID_i}, R_{ID_i}, P_{pub}, \beta_i\rangle$. If the identity ID_i already exists in the list L_{H_3} combination, $\mathcal{S}$ outputs β_i to $\mathcal{A}_I$. Otherwise, $\mathcal{S}$ chooses a random number $\beta_i \in \mathbb{Z}_q^*$, and adds the tuple $\langle au_i, ID_i, P_{ID_i}, R_{ID_i}, P_{pub}, \beta_i\rangle$ to the list L_{H_3}, then returns $H_3\left(au_i, ID_i, P_{ID_i}, R_{ID_i}, ID_o, P_{ID_o}, R_{ID_o}\right) = \beta_i$.
5) *Partial-Key-Extract-Query*: $\mathcal{S}$ holds a list L_{pp} containing tuples of the form $\langle ID_i, R_{ID_i}, d_{ID_i}\rangle$. In response to a query from $\mathcal{A}_I$ for the partial public/private key of the identity ID_i, $\mathcal{S}$ executes H_1-Query and the following steps.
   a) If $ID_i \neq ID_I$, $\mathcal{S}$ selects random numbers $\alpha_i, d_{ID_i} \in \mathbb{Z}_q^*$, and computes the partial public key $R_{ID_i} = d_{ID_i} P - \alpha_i P_{pub}$. Then $\mathcal{S}$ adds the tuple $\langle ID_i, \alpha_i, R_{ID_i}\rangle$ into the list L_{H_1}, adds the tuple $\langle ID_i, R_{ID_i}, d_{ID_i}\rangle$ into the list L_{pp}, and returns the partial public/private key $\left(R_{ID_i}, d_{ID_i}\right)$ to $\mathcal{A}_I$.
   b) Otherwise, $\mathcal{S}$ aborts (This event is denoted as E_1).
6) *Private-Key-Extract-Query*: $\mathcal{S}$ holds a list L_{ke} containing tuples of the form $\langle ID_i, x_{ID_i}, P_{ID_i}\rangle$. When $\mathcal{A}_I$ inputs the user's ID_i, $\mathcal{S}$ executes Partial-Key-Extract-Query and the following steps.
   a) If $ID_i \neq ID_I$, $\mathcal{S}$ selects random numbers $x_{ID_i} \in \mathbb{Z}_q^*$ as the secret value and computes $P_{ID_i} = x_{ID_i} P$, adds them to the tuple $\langle ID_i, x_{ID_i}, P_{ID_i}\rangle$ of the list L_{ke}. Then $\mathcal{S}$ queries the tuple

$\langle ID_i, R_{ID_i}, d_{ID_i}\rangle$ from the list L_{pp}, sets the private key $SK_{ID_i} = \left(x_{ID_i}, d_{ID_i}\right)$ and returns it to $\mathcal{A}_I$.

b) Otherwise, $\mathcal{S}$ aborts (This event is denoted as E_2).

7) *Request-Public-Key-Query*: When $\mathcal{A}_I$ queries the public key of the identity ID_i, $\mathcal{S}$ searches the lists L_{pp} and L_{ke} containing tuples of the form $\langle ID_i, R_{ID_i}, d_{ID_i}\rangle$ and $\langle ID_i, x_{ID_i}, P_{ID_i}\rangle$ respectively. $\mathcal{S}$ sets the public key $PK_{ID_i} = \left(R_{ID_i}, P_{ID_i}\right)$ and returns it to $\mathcal{A}_I$.

8) *Replace-Public-Key-Query*: $\mathcal{A}_I$ makes a replace public key query that can replace the user's public key $\left(R_{ID_i}, P_{ID_i}\right)$ with the random values $\left(R'_{ID_i}, P'_{ID_i}\right)$. Then, $\mathcal{S}$ updates the corresponding tuples $\langle ID_i, R'_{ID_i}, d_{ID_i}\rangle$ and $\langle ID_i, x_{ID_i}, P'_{ID_i}\rangle$ of the lists L_{pp} and L_{ke}, respectively.

9) *Trapdoor-Query*: $\mathcal{A}_I$ makes a trapdoor query by inputting the keyword w_i and the identity ID_i. If $ID_i = ID_I$, $\mathcal{S}$ aborts (This event is denoted as E_3). Otherwise, $\mathcal{S}$ performs the following steps to respond:

a) Selects a random number $t_{i1} \in \mathbb{Z}_q^*$. $\mathcal{S}$ searches for three tuples $\langle au_i, ID_i, P_{ID_i}, R_{ID_i}, P_{pub}, \beta_i\rangle$, $\langle ID_i, R_{ID_i}, d_{ID_i}\rangle$ and $\langle ID_i, x_{ID_i}, P_{ID_i}\rangle$ from the lists L_{H_3}, L_{pp} and L_{ke}, respectively. $\mathcal{S}$ sets $au_i = \left(x_{ID_i} + d_{ID_i}\right)\left(P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub}\right)$ and adds it to the list L_{H_3}, computes $t_{i2} = \beta_i\left(P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub}\right)$.

b) Recovers $\langle w_i, H_2(w_i), h_{i2}\rangle$ from list L_{H_2}, and sets $t_{i3} = \left(1 + \beta_i\left(x_{ID_i} + d_{ID_i}\right)\right)^{-1}\left(h_{i2} - t_{i1}\left(x_{ID_i} + d_{ID_i}\right)\right)$, then outputs the search index $T_{w_i} = (t_{i1}, t_{i2}, t_{i3})$.

10) *Challenge*: Using the identity ID^*, $\mathcal{A}_I$ first generates two challenge keywords (w_0, w_1) and sends them to $\mathcal{S}$. If $ID^* \neq ID_I$, $\mathcal{S}$ aborts (This event is denoted as E_4). Otherwise, $\mathcal{S}$ performs the following steps to respond:

a) Selects randomly $\gamma \in \{0, 1\}$, searches the tuples $\langle w_0, H_2(w_0), h_{02}\rangle$ and $\langle w_1, H_2(w_1), h_{12}\rangle$ from list L_{H_2}.

b) Sets $bP = (x_1, y_1)$, and $c_{1\gamma} = x_1$.

c) Searches the tuple $\langle au_i, ID_i, P_{ID_i}, R_{ID_i}, P_{pub}, \beta_i\rangle$. If ID_i from L_{H_3}, and computes $c_{2\gamma} = \beta_i\left(P_{ID_i} + R_{ID_i} + \alpha_i aP\right)$.

d) Selects a random number $c_{3\gamma} \in \mathbb{Z}_q^*$, and returns the ciphertext $C_\gamma = \left(c_{1\gamma}, c_{2\gamma}, c_{3\gamma}\right)$ to $\mathcal{A}_I$.

11) *Repeat-Trapdoor-Query*: $\mathcal{A}_I$ can continue to issue trapdoor query for the keyword w_i $(i \neq 0, 1)$, where this stage has the same restrictions as the trapdoor query. Let E_5 define the event that $\mathcal{A}_I$ does not query w_i.

12) *Guess*: $\mathcal{A}_I$ inputs its guess γ' to the simulator $\mathcal{S}$. It is known that $P_{pub} = aP$, $bP = (x_1, y_1)$. If $Q_i = H_3\left(au_i, ID_i, P_{ID_i}, R_{ID_i}, ID_o, P_{ID_o}, R_{ID_o}\right)\left(x_{ID_i} bP + r_{ID_i} bP + \alpha_i bP_{pub}\right)$, $\mathcal{A}_I$ will win the game, the reason is that $\mathcal{S}$ can compute $Z = \alpha_i^{-1} t_{i1}^{-1}\left(h_{i2} bP - t_{i3} Q_i - t_{i3} bP - t_{i1} b\left(P_{ID_i} + R_{ID_i}\right)\right) = abP$. Otherwise, Z is a random element in the cyclic group $\mathbb{G}$.

Analysis: We construct a simulator $\mathcal{S}$ to solve the CDH problem with the advantage ε'. It must be satisfied that the game does not abort with a high probability. If all events E_i $(i = 1, 2, 3, 4, 5)$ do not occur, then $\mathcal{S}$ does not abort. Therefore, we have

$$\begin{aligned}\Pr[\neg E] &= \Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4 \wedge \neg E_5] \\ &= \left(1 - \frac{1}{q_{H_1}}\right)^{q_{pp}+q_{ke}+q_t} \frac{1}{q_{H_1}}. \end{aligned} \tag{14}$$

Then, the advantage of $\mathcal{S}$ is $\left|\Pr\left[\gamma' = \gamma\right] - \frac{1}{2}\right|$, where

$$\begin{aligned}&\Pr\left[\gamma' = \gamma\right] \\ &= \Pr\left[\gamma' = \gamma \wedge E\right] + \Pr\left[\gamma' = \gamma \wedge \neg E\right] \\ &= \Pr\left[\gamma' = \gamma | E\right]\Pr[E] + \Pr\left[\gamma' = \gamma | E\right]\Pr[\neg E] \\ &= \frac{1}{2}(1 - \Pr[\neg E]) + \left(\varepsilon + \frac{1}{2}\right)\Pr[\neg E] \\ &= \frac{1}{2} + \varepsilon \Pr[\neg E]. \end{aligned} \tag{15}$$

Thus,

$$\left|\Pr\left[\gamma' = \gamma\right] - \frac{1}{2}\right| = \varepsilon \Pr[\neg E]. \tag{16}$$

The probability that $\mathcal{S}$ executes a hash query H_i $(i = 1, 2, 3)$ is at least $\frac{1}{q_{H_i}}$. Therefore, we have summarized the probability

$$\begin{aligned}&c_3^{-1}\left(t_3 P + t_1\left(P_{ID_u} + R_{ID_u} + \alpha_{ID_u} P_{pub}\right) + t_3 c_2\right) + c_1 c_3^{-1} t_2 \\ &= c_3^{-1}\left(t_3 P + t_1\left(x_{ID_u} P + r_{ID_u} P + s\alpha_{ID_u} P\right) + t_3\beta_{ou}\left(P_{ID_u} + R_{ID_u} + \alpha_{ID_u} P_{pub}\right)\right) + c_1 c_3^{-1} t_2 \\ &= c_3^{-1}\left(t_3 P + t_1\left(x_{ID_u} + d_{ID_u}\right) P + t_3\beta_{ou}\left(x_{ID_u} + d_{ID_u}\right) P\right) + c_1 c_3^{-1}\beta_{uo}\left(P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub}\right) \\ &= c_3^{-1}\left(t_3\left(1 + \beta_{ou}\left(x_{ID_u} + d_{ID_u}\right)\right) P + t_1\left(x_{ID_u} + d_{ID_u}\right) P\right) + c_1 c_3^{-1}\beta_{uo}\left(x_{ID_o} P + r_{ID_o} P + s\alpha_{ID_o} P\right) \\ &= c_3^{-1}\left(\left(H_2(w_i) - t_1\left(x_{ID_u} + d_{ID_u}\right)\right) P + t_1\left(x_{ID_u} + d_{ID_u}\right) P\right) + c_1 c_3^{-1}\beta_{uo}\left(x_{ID_o} + d_{ID_o}\right) P \\ &= c_3^{-1} H_2(w_i) P + c_1 c_3^{-1}\beta_{uo}\left(x_{ID_o} + d_{ID_o}\right) P \\ &= c_3^{-1}\left(H_2(w_i) + c_1\beta_{uo}\left(x_{ID_o} + d_{ID_o}\right)\right) P \\ &= k' P \\ &= \left(x'_1, y'_1\right) \end{aligned} \tag{12}$$

that $\mathcal{S}$ solves the CDH problem (and successfully guesses $\gamma' = \gamma$) as follows:

$$\varepsilon' \geq \frac{1}{q_{H_2} q_{H_3}} \varepsilon \Pr[\neg E] = \frac{\varepsilon}{q_{H_1} q_{H_2} q_{H_3}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{pp}+q_{ke}+q_t}. \tag{17}$$

Lemma 2: Assume that the adversary $\mathcal{A}_{II}$ wins the non-interactive CLPEKS scheme for IND-CLPEKS-CKA with a non-negligible advantage ε. $\mathcal{A}_{II}$ performs at most q_{H_i} $(i = 1, 2, 3)$, q_{pp}, and q_t times H_i $(i = 1, 2, 3)$-Query, Partial-Key-Extract-Query, and Trapdoor-Query, respectively. We use the ability of $\mathcal{A}_{II}$ to construct a polynomial time-bounded algorithm $\mathcal{C}$ to solve the CDH problem with an advantage

$$\varepsilon' \geq \frac{\varepsilon}{q_{H_1} q_{H_2} q_{H_3}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{pp}+q_t}. \tag{18}$$

Proof: Suppose that in the polynomial-time T, the challenger $\mathcal{C}$ first gives the generator P of a group $\mathbb{G}$. Then, $\mathcal{C}$ chooses two random numbers $a, b \in \mathbb{Z}_q^*$. Finally, $\mathcal{C}$ is given an instance (aP, bP) to compute the CDH problem, and constructs a simulator $\mathcal{S}$ to compute $cP = abP \bmod n$. ■

1) *Setup*: $\mathcal{S}$ runs Setup, generates a random number s as the master key, and then sets $P_{pub} = sP$ and $P_{ID_i} = aP$. In addition, $\mathcal{S}$ randomly chooses ID_I as the challenger identity. Finally, $\mathcal{S}$ generates system parameters $params = \{E(\mathbb{F}_q), q, \mathbb{G}, P, P_{pub}, H_1, H_2, H_3\}$, sends P_{ID_i} and $params$ to adversary $\mathcal{A}_{II}$. $\mathcal{S}$ responds to query of $\mathcal{A}_{II}$ as follows.
2) H_1-*Query*: $\mathcal{S}$ maintains a tuple $\langle ID_i, \alpha_i, R_{ID_i}\rangle$ in the list L_{H_1}. $\mathcal{A}_{II}$ queries $H_1(ID_i)$, if the identity ID_i already exists in the list L_{H_1} combination, $\mathcal{S}$ outputs α_i to $\mathcal{A}_{II}$. Otherwise, $\mathcal{S}$ selects a random number $\alpha_i \in \mathbb{Z}_q^*$, and adds the tuple $\langle ID_i, \alpha_i, R_{ID_i}\rangle$ to the list L_{H_1}, then returns α_i to $\mathcal{A}_{II}$.
3) H_2-*Query*: $\mathcal{S}$ maintains a tuple $\langle w_i, H_2(w_i), h_{i2}\rangle$ in the list L_{H_2}. $\mathcal{A}_{II}$ queries $H_2(w_i)$, if the identity ID_i already exists in the list L_{H_2} combination, $\mathcal{S}$ outputs h_{i2} to $\mathcal{A}_{II}$. Otherwise, $\mathcal{S}$ chooses a random number $h_{i2} \in \mathbb{Z}_q^*$, and adds the tuple $\langle w_i, H_2(w), h_{i2}\rangle$ to the list L_{H_2}, then returns h_{i2} to $\mathcal{A}_{II}$.
4) H_3-*Query*: $\mathcal{S}$ maintains a list L_{H_3} with tuples $\langle au_i, ID_i, P_{ID_i}, R_{ID_i}, P_{pub}, \beta_i\rangle$. If the identity ID_i already exists in the list L_{H_3} combination, $\mathcal{S}$ outputs β_i to $\mathcal{A}_{II}$. Otherwise, $\mathcal{S}$ chooses a random number $\beta_i \in \mathbb{Z}_q^*$, and adds the tuple $\langle au_i, ID_i, P_{ID_i}, R_{ID_i}, P_{pub}, \beta_i\rangle$ to L_{H_3}, then outputs β_i.
5) *Partial-Key-Extract-Query*: $\mathcal{S}$ holds a list L_{pp} containing tuples of the form $\langle ID_i, R_{ID_i}, d_{ID_i}\rangle$. When $\mathcal{A}_{II}$ inputs the user's ID_i, $\mathcal{S}$ executes the following steps:
   a) If $ID_i \neq ID_I$, $\mathcal{S}$ selects random numbers $r_{ID_i} \in \mathbb{Z}_q^*$ and computes the partial public key $R_{ID_i} = r_{ID_i}P$, and the partial private key $d_{ID_i} = r_{ID_i} + s\alpha_i$. Then $\mathcal{S}$ respectively adds R_{ID_i} to L_{H_3} and d_{ID_i} to L_{pp}, returns (R_{ID_i}, d_{ID_i}) to $\mathcal{A}_{II}$.
   b) Otherwise, $\mathcal{S}$ aborts (This event is denoted as E_1).
6) *Trapdoor-Query*: $\mathcal{A}_{II}$ makes a trapdoor query by inputting the keyword w_i and the identity ID_i. If $ID_i = ID_I$, $\mathcal{S}$ aborts (This event is denoted as E_2). Otherwise, $\mathcal{S}$ performs the following steps to respond:
   a) Selects $v, t_{i1} \in \mathbb{Z}_q^*$ randomly, searching for the lists L_{H_3} and L_{pp}, respectively. $\mathcal{S}$ computes $au_i = (v + d_{ID_i})(P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub})$ and adds it to the list L_{H_3}, sets $t_{i2} = \beta_i (P_{ID_o} + R_{ID_o} + \alpha_{ID_o} P_{pub})$.
   b) Recovers the tuple containing the form $\langle w_i, H_2(w_i), h_{i2}\rangle$ from the list L_{H_2}, computes $t_{i3} = (1 + \beta_i (v + d_{ID_i}))^{-1} (h_{i2} - t_{i1}(v + d_{ID_i}))$, and outputs the search index $T_{w_i} = (t_{i1}, t_{i2}, t_{i3})$.
7) *Challenge*: $\mathcal{A}_{II}$ first generates two challenge keywords (w_0, w_1) with identity ID^* and sends them to $\mathcal{S}$. If $ID^* \neq ID_I$, $\mathcal{S}$ aborts (This event is denoted as E_3). Otherwise, $\mathcal{S}$ performs the following steps to respond:
   a) Selects randomly $\gamma \in \{0, 1\}$, and searches the tuples $\langle w_0, H_2(w_0), h_{02}\rangle$ and $\langle w_1, H_2(w_1), h_{12}\rangle$ from list L_{H_2}.
   b) Computes $bP = (x_1, y_1)$, let $c_{1\gamma} = x_1$.
   c) Lets $au_{ou} = au_i$. Then, $\mathcal{S}$ computes $c_{2\gamma} = \beta_i (P_{ID_i} + R_{ID_i} + \alpha_i sP)$.
   d) Selects a random number $c_{3\gamma} \in \mathbb{Z}_q^*$, and returns the ciphertext $C_\gamma = (c_{1\gamma}, c_{2\gamma}, c_{3\gamma})$ to $\mathcal{A}_{II}$.
8) *Repeat-Trapdoor-Queries*: $\mathcal{A}_{II}$ can continue to issue trapdoor queries for the keyword w_i $(i \neq 0, 1)$, where this stage has the same restrictions as trapdoor queries. (Let E_4 define the event that $\mathcal{A}_{II}$ does not query w_i).
9) *Guess*: At the end, the adversary $\mathcal{A}_{II}$ inputs its guess γ' to the simulator $\mathcal{S}$. It is known that $P_{ID_i} = aP$, $bP = (x_1, y_1)$. If $Q_i = vbP + bP_{ID_i} + bR_{ID_i} + \alpha_i sbP$, $\mathcal{S}$ easily computes $Z = t_{i1}^{-1}(t_{i3}bP + t_{i3}\beta_i b(vP + r_{ID_i}P + \alpha_i sP) + t_{i1}Q_i - h_{i2}bP) = abP$, then $\mathcal{A}_{II}$ wins the game. Otherwise, Z is a random element in the cyclic group $\mathbb{G}$.

Analysis. We make a simulator $\mathcal{S}$ to solve the CDH problem with the advantage ε'. It must be satisfied that the game does not abort with a high probability. If all events E_i $(i = 1, 2, 3, 4)$ do not occur, then $\mathcal{S}$ does not abort. Thus, we have

$$\begin{aligned}\Pr[\neg E] &= \Pr[\neg E_1 \wedge \neg E_2 \wedge \neg E_3 \wedge \neg E_4] \\ &= \left(1 - \frac{1}{q_{H_1}}\right)^{q_{pp}+q_t} \frac{1}{q_{H_1}}.\end{aligned} \tag{19}$$

Next, the advantage of $\mathcal{S}$ is $\left|\Pr[\gamma' = \gamma] - \frac{1}{2}\right|$, where

$$\begin{aligned}&\Pr[\gamma' = \gamma] \\ &= \Pr[\gamma' = \gamma \wedge E] + \Pr[\gamma' = \gamma \wedge \neg E] \\ &= \Pr[\gamma' = \gamma | E] \Pr[E] + \Pr[\gamma' = \gamma | E] \Pr[\neg E] \\ &= \frac{1}{2}(1 - \Pr[\neg E]) + \left(\varepsilon + \frac{1}{2}\right) \Pr[\neg E] \\ &= \frac{1}{2} + \varepsilon \Pr[\neg E].\end{aligned} \tag{20}$$

TABLE 2. Comparison of security properties.

Scheme	Security properties					
	Pf	Nc	Nk	Au	OKGA	IKGA
[11]	×	√	×	√	√	√
[13]	×	√	√	√	√	√
[15]	√	√	√	√	√	√
[31]	×	√	√	×	√	×
[33]	√	√	√	√	×	×
[34]	√	√	√	√	√	√
[35]	√	√	√	√	×	×
[36]	√	√	√	√	√	√
Ours	√	√	√	√	√	√

√−supported, ×−not supported

TABLE 3. Running time of operators (ms).

Notation	Definition	Time
T_h	The time cost of a general hash function	0.007
T_{hp}	The time cost of a hash function map-to-point	5.493
T_{pm}	The time cost of a scalar point multiplication	2.165
T_{pa}	The time cost of a scalar point addition	0.013
T_{bp}	The time cost of a bilinear pairing	5.427

Hence,

$$\left|\Pr\left[\gamma' = \gamma\right] - \frac{1}{2}\right| = \varepsilon \Pr\left[\neg E\right], \tag{21}$$

where ε is non-negligible. $\mathcal{S}$ has at least $\frac{1}{q_{H_i}}$ probability of performing the hash query $H_i\,(i = 1, 2, 3)$. Therefore, we have summarized the probability that $\mathcal{S}$ solves the CDH problem as follows:

$$\varepsilon' \geq \frac{1}{q_{H_2} q_{H_3}} \varepsilon \Pr\left[\neg E\right] = \frac{\varepsilon}{q_{H_1} q_{H_2} q_{H_3}} \left(1 - \frac{1}{q_{H_1}}\right)^{q_{pp}+q_t}. \tag{22}$$

VI. PERFORMANCE ANALYSIS

In this section, we will analyze the performance of the proposed CLPEKS scheme from three aspects: security property, computational cost, and communication cost. We will compare the results of our scheme with [11], [13], [15], [31], [33], [34], [35], and [36].

A. SECURITY PROPERTY

We denote pairing-free, no certificate, no key escrow, authentication, offline keyword guessing attacks, and inside keyword guessing attacks as Pf, Nc, Nk, Au, OKGA, and IKGA, respectively. We can see from Table 2 that [11], [13], and [31] all depend on the costly bilinear pair. Reference [11] has the key escrow problem. Reference [31] has identity authentication flaws during cloud testing. References [33] and [35] cannot resist OKGA. References [31], [33], and [35] cannot resist IKGA attack. References [15], [34], and [36] and our scheme satisfy all security properties.

B. COMPUTATIONAL COST

In our scheme, we use the execution results in [13] to simulate and estimate the running time. The simulation environment is a Dell computer with an i5-4460S 2.90Hz processor, 4GB memory, and Windows 8 OS, and it uses the MIRACL library. The Tate pairing operation defined over the supersingular elliptic curve $E\left(\mathbb{F}_q\right) : y^2 = x^3 + x$ with a finite field $\mathbb{F}_q$ of order p is used. In Table 3, we excerpt the definition and running time of the operators, the symbols T_h, T_{hp}, T_{pm}, T_{pa}, and T_{bp} respectively denote the general hash function operation, the hash function mapping to point operation, the scalar point multiplication operation, the scalar point addition operation, and the bilinear pairing operation. Their running times are 0.007 ms, 5.493 ms, 2.165 ms, 0.013 ms, and 5.427 ms, respectively.

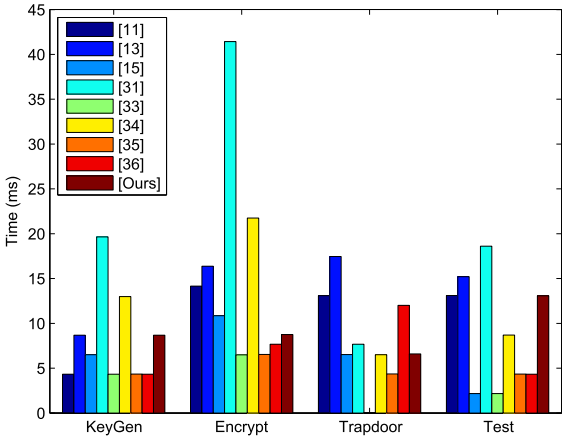

FIGURE 3. The comparison of computational cost.

We have summarized the computational cost of the nine schemes in Table 4 and Fig. 3. In the KeyGen algorithm, our scheme has the same running time as [13], which is 55.85% and 33.23% less than the running time of [31] and [34], respectively. The Encrypt algorithm of our scheme is 34.75%, 34.07% and 14.08% more than the running time of [33], [35], and [36], respectively. The Trapdoor algorithm of our scheme is 1%, 1.21% and 51.22% more than the running time of [15], [34], and [35], respectively. Compared with [11], [13], and [31], the running time of our scheme in the test algorithm, is reduced by 0.31%, 14.08%, and 29.78%, respectively.

The key generation algorithm and Test algorithms are usually performed by KGC and cloud servers, whose resource is considered adequate. Reference [33] cannot resist IKGA and OKGA. The total time consumption of the Encrypt algorithm and Trapdoor algorithm of [15], [34], and [36], and ours is 17.381 ms, 28.262 ms, 19.674 ms, and 15.339 ms. Therefore, our algorithm has a better computational cost for data owners and users with limited resources.

C. COMMUNICATION COST

For the communication cost, we will give the comparison results of the nine schemes. We use the symbols PK, CT, and TD to denote the sizes of the public key, the ciphertext, and the trapdoor. To achieve an 80-bit security level, we have large prime numbers p and q of 160 bits and 512 bits, respectively. The length of the elements in $\mathbb{G}$ and $\mathbb{G}_T$ are 1024 bits and 512 bits, respectively. At the same security level, we assume that the length of the element in $\mathbb{Z}_q^*$ is 160 bits. Table 5 and Fig. 4 show that for the PK, the communication cost of the proposed scheme is 50% higher than that of [11] and [36].

TABLE 4. Computational cost comparison (ms).

Scheme	KeyGen	Encrypt	Trapdoor	Test
[11]	$2T_{pm} = 4.33$	$T_{hp} + 4T_{pm} = 14.153$	$T_{hp} + T_{pm} + T_{bp} = 13.085$	$T_{pm} + 2T_{bp} = 13.109$
[13]	$2T_h + 4T_{pm} = 8.674$	$T_{hp} + 2T_h + 5T_{pm} + 3T_{pa} = 16.371$	$T_{hp} + 2T_h + 3T_{pm} + 2T_{pa} + T_{bp} = 17.455$	$2T_{pm} + 2T_{pa} + 2T_{bp} = 15.21$
[15]	$T_h + 3T_{pm} = 6.502$	$3T_h + 5T_{pm} + T_{pa} = 10.859$	$2T_h + 3T_{pm} + T_{pa} = 6.522$	$T_{pm} + T_h = 2.172$
[31]	$2T_{hp} + 4T_{pm} = 19.646$	$3T_{hp} + T_h + 4T_{pm} + T_{pa} + 3T_{bp} = 41.433$	$T_{hp} + T_{pm} + T_{pa} = 7.671$	$2T_{hp} + T_h + T_{pm} + 2T_{pa} + T_{bp} = 18.611$
[33]	$2T_{pm} = 4.33$	$3T_{pm} = 6.495$	$--$	$T_{pm} = 2.165$
[34]	$6T_{pm} = 12.99$	$10T_{pm} + 8T_{pa} = 21.754$	$3T_{pm} + T_{pa} = 6.508$	$3T_{pa} + 4T_{pm} = 8.699$
[35]	$2T_h + 2T_{pm} = 4.344$	$T_h + 3T_{pm} + 2T_{pa} = 6.528$	$2T_{pm} + 2T_{pa} = 4.356$	$T_h + 2T_{pm} = 4.337$
[36]	$2T_{pm} = 4.33$	$2T_h + T_{pm} + T_{hp} = 7.672$	$2T_h + 3T_{pm} + T_{hp} = 12.002$	$2T_{pm} = 4.33$
Ours	$2T_h + 4T_{pm} = 8.674$	$2T_h + 4T_{pm} + 6T_{pa} = 8.752$	$2T_h + 3T_{pm} + 6T_{pa} = 6.587$	$6T_{pm} + 5T_{pa} = 13.068$

TABLE 5. Communication cost comparison (bits).

Scheme	Size of PK	Size of CT	Size of TD
[11]	$\|\mathbb{G}\| = 1024$	$\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 1184$	$\|\mathbb{G}_T\| = 512$
[13]	$2\|\mathbb{G}\| = 2048$	$2\|\mathbb{G}\| = 2048$	$\|\mathbb{G}_T\| = 512$
[15]	$2\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 2208$	$2\|\mathbb{G}\| = 2048$	$\|\mathbb{Z}_q^*\| = 160$
[31]	$2\|\mathbb{G}\| = 2048$	$\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 1184$	$\|\mathbb{G}\| = 1024$
[33]	$2\|\mathbb{G}\| = 2048$	$\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 1184$	$\|\mathbb{Z}_q^*\| = 160$
[34]	$6\|\mathbb{G}\| = 6144$	$2\|\mathbb{G}\| = 2048$	$2\|\mathbb{G}\| = 2048$
[35]	$2\|\mathbb{G}\| = 2048$	$\|\mathbb{G}\| + 3\|\mathbb{Z}_q^*\| = 1504$	$\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 1184$
[36]	$\|\mathbb{G}\| = 1024$	$\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 1184$	$\|\mathbb{G}\| + \|\mathbb{Z}_q^*\| = 1184$
Ours	$2\|\mathbb{G}\| = 2048$	$\|\mathbb{G}\| + 2\|\mathbb{Z}_q^*\| = 1344$	$\|\mathbb{G}\| + 2\|\mathbb{Z}_q^*\| = 1344$

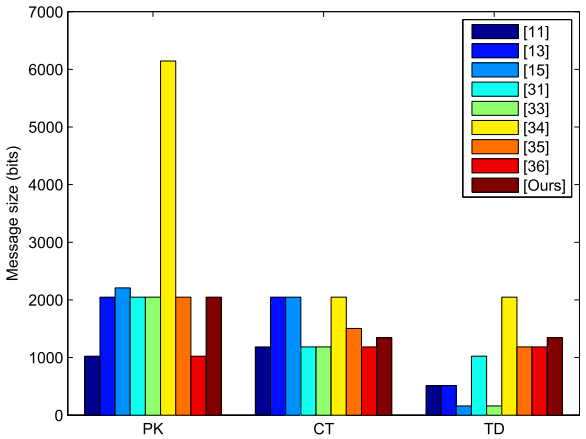

FIGURE 4. The comparison of communication cost.

For CT, the communication cost of the proposed scheme is 13.51% higher than that of [11], [31], [33], and [36]. For TD, the communication cost of [11], [13], [15], [31], [33], [35], and [36] are 61.90%, 61.90%, 88.1%, 23.81%, 88.1%, 11.90%, and 11.90% lower than ours, respectively. Therefore, the proposed scheme has no obvious advantage in terms of communication cost.

VII. CONCLUSION

In this paper, we propose a pairing-free CLPEKS scheme for the IIoT. It does not have the problems of certificate management and key escrow. Furthermore, the scheme is proved to be secure in the random oracle model. The performance analysis shows that our algorithm has a better overall performance, even though it has a high communication cost. Thus, the proposed scheme is more suitable for the actual IIoT. In future work, we will investigate how to authorize multiple users to perform searchable encryption, and how to ensure efficient data sharing.

REFERENCES

[1] A. Sadeghi, C. Wachsmann, and M. Waidner, "Security and privacy challenges in industrial Internet of Things," in *Proc. 52nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, San Francisco, CA, USA, Jun. 2015, pp. 1–6.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[3] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, Y. Xu, S. Srivastav, J. Wu, H. Simitci, and J. Haridas, "Windows Azure storage: A highly available cloud storage service with strong consistency," in *Proc. 23rd ACM Symp. Operating Syst. Princ.*, Cascais, Portugal, 2011, pp. 143–157.

[4] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of 'big data' on cloud computing: Review and open research issues," *Inf. Syst.*, vol. 47, pp. 98–115, Jan. 2015.

[5] L. Wei, H. Zhu, Z. Cao, X. Dong, W. Jia, Y. Chen, and A. V. Vasilakos, "Security and privacy for storage and computation in cloud computing," *Inf. Sci.*, vol. 258, pp. 371–386, Feb. 2014.

[6] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy.*, Berkeley, CA, USA, Mar. 2000, pp. 44–55.

[7] C. Bösch, P. Hartel, W. Jonker, and A. Peter, "A survey of provably secure searchable encryption," *ACM Comput. Surveys*, vol. 47, no. 2, pp. 1–51, Jan. 2015.

[8] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.*, Interlaken, Switzerland, 2004, pp. 506–522.

[9] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" *Comput. Commun.*, vol. 32, no. 2, pp. 394–396, Feb. 2009.

[10] W.-C. Yau, S.-H. Heng, and B.-M. Goi, "Off-line keyword guessing attacks on recent public key encryption with keyword search schemes," in *Proc. Int. Conf. Autonomic Trusted Comput.*, Oslo, Norway, 2008, pp. 100–105.

[11] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vols. 403–404, pp. 1–14, Sep. 2017.

[12] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Workshop Theory Appl. Cryptograph. Techn.*, Paris, France, 1984, pp. 47–53.

[13] 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.

[14] Y. Lu and J. Li, "Lightweight public key authenticated encryption with keyword search against adaptively-chosen-targets adversaries for mobile devices," *IEEE Trans. Mobile Comput.*, vol. 21, no. 12, pp. 4397–4409, Dec. 2022.
[15] D. Shiraly, N. Pakniat, M. Noroozi, and Z. Eslami, "Pairing-free certificateless authenticated encryption with keyword search," *J. Syst. Archit.*, vol. 124, Mar. 2022, Art. no. 102390.
[16] C. Dong, G. Russello, and N. Dulay, "Shared and searchable encrypted data for untrusted servers," *J. Comput. Secur.*, vol. 19, no. 3, pp. 367–397, May 2011.
[17] H. S. Rhee, W. Susilo, and H.-J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electron. Exp.*, vol. 6, no. 5, pp. 237–243, 2009.
[18] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. Workshop Secure Data Manage.*, Seoul, South Korea, 2006, pp. 75–83.
[19] 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," *J. Syst. Softw.*, vol. 83, no. 5, pp. 763–771, May 2010.
[20] B. Wang, T. Chen, and F. Jeng, "Security improvement against malicious server's attackfor a dPEKS scheme," *Int. J. Inf. Educ. Technol.*, vol. 1, no. 4, pp. 350–353, 2011.
[21] Q. Tang and L. Chen, "Public-key encryption with registered keyword search," in *Proc. Eur. Public Key Infrastructure Workshop*, Pisa, Italy, 2009, pp. 163–178.
[22] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack," *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2013.
[23] C. Liu, L. Zhu, M. Wang, and Y.-A. Tan, "Search pattern leakage in searchable encryption: Attacks and new construction," *Inf. Sci.*, vol. 265, pp. 176–188, May 2014.
[24] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 789–798, Apr. 2016.
[25] Y. Lu, J. Li, and Y. Zhang, "Secure channel free certificate-based searchable encryption withstanding outside and inside keyword guessing attacks," *IEEE Trans. Services Comput.*, vol. 14, no. 6, pp. 2041–2054, Nov. 2021.
[26] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," *J. Cryptol.*, vol. 21, no. 3, pp. 350–391, Jul. 2008.
[27] H. Li, Q. Huang, J. Shen, G. Yang, and W. Susilo, "Designated-server identity-based authenticated encryption with keyword search for encrypted emails," *Inf. Sci.*, vol. 481, pp. 330–343, May 2019.
[28] B. Qin, Y. Chen, Q. Huang, X. Liu, and D. Zheng, "Public-key authenticated encryption with keyword search revisited: Security model and constructions," *Inf. Sci.*, vol. 516, pp. 515–528, Apr. 2020.
[29] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Taipei, Taiwan, 2003, pp. 452–473.
[30] P. Yanguo, C. Jiangtao, P. Changgen, and Y. Zuobin, "Certificateless public key encryption with keyword search," *China Commun.*, vol. 11, no. 11, pp. 100–113, Nov. 2014.
[31] M. Ma, D. He, N. Kumar, 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.
[32] Y. Zhang, X. Liu, X. Lang, Y. Zhang, and C. Wang, "VCLPKES: Verifiable certificateless public key searchable encryption scheme for industrial Internet of Things," *IEEE Access*, vol. 8, pp. 20849–20861, 2020.
[33] Y. Lu and J.-G. Li, "Constructing pairing-free certificateless public key encryption with keyword search," *Frontiers Inf. Technol. Electron. Eng.*, vol. 20, no. 8, pp. 1049–1060, Aug. 2019.
[34] M. Ma, M. Luo, S. Fan, and D. Feng, "An efficient pairing-free certificateless searchable public key encryption for cloud-based IIoT," *Wireless Commun. Mobile Comput.*, vol. 2020, pp. 1–11, Dec. 2020.
[35] 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.
[36] M. R. Senouci, I. Benkhaddra, A. Senouci, and F. Li, "A provably secure free-pairing certificateless searchable encryption scheme," *Telecommun. Syst.*, vol. 80, no. 3, pp. 383–395, Jul. 2022.
[37] Z. Hu, L. Deng, Y. Wu, H. Shi, and Y. Gao, "Secure and efficient certificateless searchable authenticated encryption scheme without random Oracle for industrial Internet of Things," *IEEE Syst. J.*, vol. 17, no. 1, pp. 1304–1315, Mar. 2023.

XIAOGUANG LIU received the Ph.D. degree in mathematics from Shaanxi Normal University, Xi'an, China, in 2014. From 2017 to 2020, he was a Postdoctoral Fellow with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China. He is currently a Lecturer with the School of Mathematics, Southwest Minzu University, Chengdu. His recent research interests include cryptography, network security, and optimization computation.

HAO DONG received the B.S. degree in electronic and information engineering from Hankou University, in 2018. He is currently pursuing the master's degree with the School of Computer Science and Engineering, Southwest Minzu University (SMU), Chengdu, China. His current research interests include information security and blockchain.

NEHA KUMARI received the B.Tech. degree in CSE, in 2015, and the M.Tech. degree in cyber security, in 2017. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The LNM Institute of Information Technology (LNMIIT), Jaipur, Rajasthan. She qualified UGC-NET, in 2018. She is an active member of the Centre for Cryptology, Cybersecurity and Digital Forensics, LNMIIT. Her research interests include blockchain, cryptographic protocols/primitives using elliptic curves, and pairing-based cryptography.

JAYAPRAKASH KAR (Senior Member, IEEE) received the M.Sc. and M.Phil. degrees in mathematics from Sambalpur University and the M.Tech. and Ph.D. degrees in computer science (cryptographic protocols) from Utkal University, India. He is currently a Visiting Researcher with the Department of Electrical and Computer Engineering, University of Waterloo, Canada. He is also an Associate Professor and the HoD of the Department of Computer Science and Engineering, The LNM Institute of Information Technology, Jaipur, India. He is the Center-Lead of the Center for Cryptology, Cybersecurity and Digital Forensics, LNMIIT. His current research interests include cryptographic protocols and primitives using elliptic curves and pairing-based cryptography in random oracle and standard models, zero-knowledge proofs, secret sharing, and multi-party computation. He is a Life Member of the International Association for Cryptology Research (IACR) and the Cryptology Research Society of India and an Associate Member of ACM, the International Association of Computer Science and Information Technology (Singapore), and the International Association of Engineers (USA).