Journal of Information Security and Applications 53 (2020) 102525

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Certificateless authenticated encryption with keyword search: Enhanced security model and a concrete construction for industrial IoT

Nasrollah Pakniat[a,*], Danial Shiraly[b], Ziba Eslami[b]

[a] Information Science Research Center, Iranian Research Institute for Information Science and Technology (IranDoc), Tehran, Iran
[b] Department of Data and Computer Science, Shahid Beheshti University, G. C., Tehran, Iran

ARTICLE INFO

Article history:
Available online 1 May 2020

Keywords:
Certificateless
Searchable encryption
Authenticated encryption
Keyword guessing attack
Industrial IoT

ABSTRACT

The Industrial Internet of Things (IIoT) represents a variety of IoT applications in the industrial sector of economy and is heavily underpinned by computational capabilities of cloud computing to reduce the cost of on-demand services. However, cloud computing does not protect the privacy of the outsourced data unless some form of encryption is applied. Recently, some Certificateless Authenticated Encryption with Keyword Search (CLAEKS) schemes have been proposed. However, it is known in the literature that the security of these schemes is proved based on a weak security model. In this paper, we first provide a more powerful and realistic security model for CLAEKS schemes. Then, we propose a new CLAEKS scheme and prove its security in the enhanced security model. The comparison results show that the proposed CLAEKS scheme is the only existing searchable encryption scheme in the certificateless setting that is secure in the enhanced security model.

1. Introduction

The internet of things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network in a wide range of applications without requiring human-to-human or human-to-computer interaction. A popular branch of IoT is the Industrial IoT (IIoT) which is the application of IoT in the industrial sector. According to the Gartner, by 2020, approximately 63 million of IoT devices will be connected to the enterprise network in each second [1] and the global economic impact of IoT reaches to 2 trillion USD [2].

Cloud computing has the advantages of unlimited storage space, fast computing, high service availability and low cost. These advantages, makes cloud storage a good media to store the huge IIoT data. A typical network architecture for IIoT data storage is depicted in Fig. 1. In such an architecture, the collected data by the sensors will be sent to the cloud server over the Internet. However, a main problem here is the confidentiality of the data i.e., the data needs to be securely stored on the cloud so that only authorized users can access the collected data. Unfortunately, the cloud servers are not completely trusted and outsourcing data to them may result in loss of privacy. Therefore, sensitive data must be encrypted before outsourcing. While encryption can protect the privacy and confidentiality of data, it destroys the ability to search over the data and eliminates the ability to share them. Searchable public key encryption is the solution provided by cryptography to these problems. It is a branch of searchable encryption [3] that preserves the confidentiality of the encrypted data, and maintains the search and sharing ability over them [4–6]. The first searchable public key encryption, called Public key Encryption with Keyword Search (PEKS), was proposed by Boneh et al. in [4]. In [5], the authors noticed that in order to maintain security, this scheme needs a secure channel to transmit the ciphertexts to the server and improved the definition of PEKS to remove this limitation. In [7,8], the authors introduced an important attack against PEKS schemes called Keyword Guessing Attack (KGA). In a KGA, after intercepting a trapdoor, the attacker generates the searchable ciphertexts corresponding to possible keywords and searches over them using the intercepted trapdoor [9]. The feasibility of the attack comes from the fact that the keyword space is small. Therefore, the attack will determine the corresponding keyword to the intercepted trapdoor uniquely. To propose a solution against KGA, in [6], Rhee et al. modified the definition of PEKS and defined the concept of "searchable public key encryption with a designated tester" (dPEKS for short) which provides a new security property called trapdoor indistinguishability. Informally, trapdoor indistinguishability means that no polynomial time adversary can distinguish the trapdoor corresponding to one keyword from the trap-

* Corresponding author.
E-mail addresses: pakniat@irandoc.ac.ir (N. Pakniat), d.shiraly@mail.sbu.ac.ir (D. Shiraly), z_eslami@sbu.ac.ir (Z. Eslami).

https://doi.org/10.1016/j.jisa.2020.102525
2214-2126/

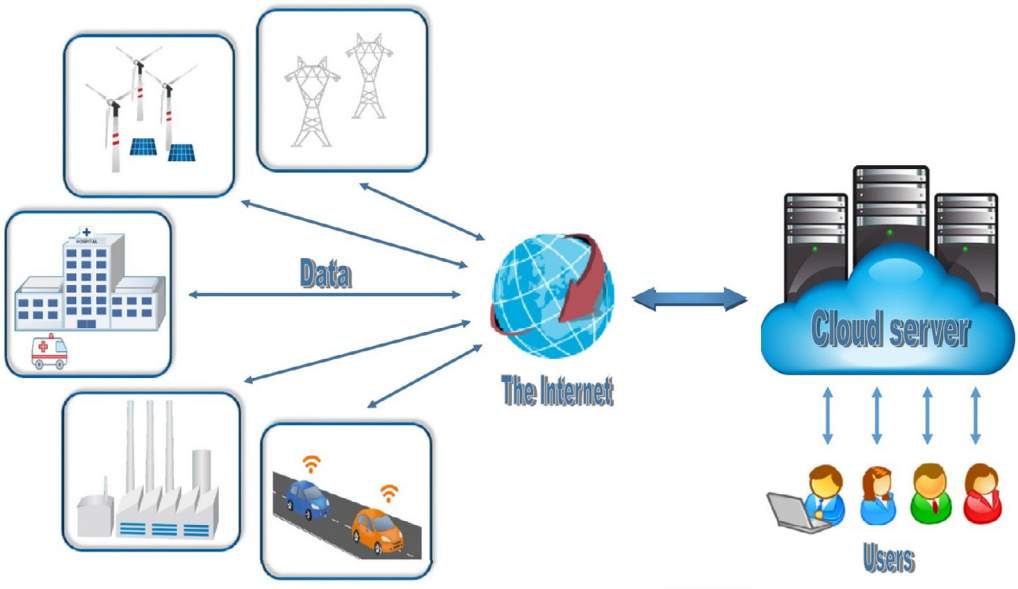

Fig. 1. A typical network architecture for IIoT data storage.

door corresponding to another. Furthermore, the authors proved that providing the new security requirement is equal to being secure against KGA. After Rhee et al.'s work, a lot of dPEKS schemes were proposed [10–15]. However, these schemes only provide security against KGAs performed by outside attackers (anyone except the receiver and the server). To consider security against KGAs performed by inside attackers (i.e., the server), in [16], the authors proposed a chaos based Public key Authenticated Encryption with Keyword Search (PAEKS) scheme. However, this scheme is proven to be insecure by Noroozi et al. [17]. In [18], the authors formally defined the concept of PAEKS and its security model. In [19], Noroozi and Eslami noticed some drawbacks in the security model provided in [18] and presented an enhanced security model for PAEKS. The authors further improved the scheme of [18] to be secure in the enhanced security model. In 2020, Qin et al. [20] considered a further enhanced security model for PAEKS schemes and proposed a PAEKS scheme with provable security in the new enhanced security model. In PAEKS, both the sender and the receiver are equipped with private and public key pairs. The sender uses his private key together with the public key of the receiver in the generation of the ciphertexts. The receiver, in order to search over the encrypted data, uses his private key together with the public key of the sender to generate the proper trapdoors. In this way, the generated trapdoors are only applicable to ciphertexts generated by the intended sender and therefore, PAEKS schemes are secure against KGAs performed by both inside and outside attackers.

Unfortunately, all the above mentioned schemes suffer from the certificate management problem that exists in public key setting. Therefore, searchable encryption schemes in identity-based cryptography (denoted by (IBEKS)) were proposed first in [21]. To overcome KGA in identity-based setting, some designated tester IBEKS (dIBEKS) were proposed in [14,22,23]. However, Noroozi et al. in [24] showed that all the existing dIBEKS schemes suffer from some sort of security flaw and are hence insecure. In [25], the authors extended the concept of PAEKS to the identity-based cryptography setting and proposed two authenticated encryption with keyword search schemes.

In identity-based cryptography, there exists a trusted entity known as a Key Generation Center (KGC) who is responsible for generating users' private keys. As a result, the certificate management problem in the public-key setting is actually replaced by the so-called key escrow problem in identity-based setting. To simultaneously avoid both the certificate management and the key escrow problems, searchable encryption schemes in the certificateless cryptography setting [26] can be used where KGC partially generates private keys.

In [27], Peng et al. proposed the first Certificateless Encryption with Keyword Search (CLEKS) scheme. However, in [28], it is shown that Peng et al.'s scheme is not secure against malicious KGCs. In [29] and [30], two other CLEKS schemes were proposed. These schemes provide the security of ciphertexts however, they are not practical since they require secure channels for communications between senders/receivers and cloud servers. In addition to being inefficient, these schemes are also insecure against KGA the same as all ordinary PEKS schemes. To provide security against KGAs performed by outside attackers, in [31,32], the authors proposed a designated tester CLEKS (dCLEKS) scheme. However, in [31], the server needs to know the searched keywords in order to carry out the search. As a consequence, this scheme does not satisfy the minimum security requirement of searchable encryption schemes, i.e., the privacy of keywords.

1.1. Motivation and contribution

Recently, in [33,34], the authors proposed two Certificateless authenticated encryption with keyword search (CLAEKS) schemes. Unfortunately, these schemes suffer from the following serious drawbacks which destroy almost all the potentials of the defined concept:

- An important drawback, reported first in [17] for PEKS setting, is that a data owner can generate searchable ciphertexts corresponding to only one receiver. By generating searchable ciphertexts for more than one receiver, each of the data receivers are able to search over the data encrypted for the other ones as well. Liu et al. in [35] demonstrated that [33] suffers from this drawback in CLAEKS setting.
- The second drawback, which is also considered already for the case of PEKS by researchers in [20], is that the trapdoor corresponding to a keyword can be computed only by observing the corresponding ciphertext to that keyword. Therefore, while the CLAEKS schemes of [33–35] are presented as probabilistic

schemes, they act like deterministic ones. In [36], it is shown that all the existing CLAEKS schemes suffer from this problem.

We noticed that these drawbacks are due to the weakly defined security models in [33–35]. In more details, the defined security model in [33,34] has the following limitations (although the security model of [35] only suffers from the second limitation):

- The input of the Trapdoor oracle is only the keyword. In other words, the data owner and the receiver are fixed in their models and attackers are not allowed to obtain trapdoors corresponding to a pair of the (data owner, receiver) different from the specified ones in the security model. Therefore, CLAEKS schemes which are proven secure in this model, only guarantee security when a data owner generates searchable ciphertexts for only one receiver. Obviously, this security model doesn't capture full potentials of public key cryptography.
- The security models of [33–35] don't provide access to the CLAEKS oracle for the challenged keywords. In other words, they don't allow attackers to obtain searchable ciphertexts corresponding to any keyword of their choices. However, as in the case of encryption schemes, a well defined security model for CLAEKS, should give attackers oracle access to CLAEKS algorithm for any keyword of its choice.

In this paper, we first target the security model deficiencies and present an enhanced security model for CLAEKS schemes. Then, we propose a new efficient CLAEKS scheme for IIoT and prove its security in the enhanced security model under CDH and GBDH assumptions. The comparison results show that the proposed CLAEKS scheme is the only existing searchable encryption scheme in the certificateless setting that provides the needed security requirements. The analysis of computational and communication complexities show that the obtained security is achieved at some affordable computational cost in the trapdoor generation.

1.2. Paper organization

The rest of this paper is organized as follows: Section 2 reviews preliminary material including bilinear group description and some mathematical problems. In Sections 3, we review the concept of certificateless authenticated encryption with keyword search (CLAEKS) and the formal definition of its security model. In Section 4, we describe the proposed CLAEKS scheme. In Section 5, we analyze the security of the proposed scheme and in Section 6, its performance is analyzed. Finally, conclusions are provided in Section 7.

2. Preliminaries

In this section, we provide a brief review of preliminary materials including bilinear group description and some mathematical problems. The details of this section is mostly from Eslami and Pakniat [37].

Definition 1. A symmetric bilinear group description Γ is a tuple (q, G_1, G_2, e, P) where G_1 and G_2 are two groups of the same prime order q, e: $G_1 \times G_1 \rightarrow G_2$ is an efficiently computable non-degenerate bilinear map, and P is the generator of G_1.

Definition 2. Given a symmetric bilinear group description Γ, it is said that the gap bilinear Diffie-Hellman (*GBDH*) assumption holds if the advantage of any probabilistic polynomial time (PPT) adversary A as defined below is negligible.

$$Adv_{\Gamma}^{GBDH}(A, q_{DBDH}) := Pr[T = e(P,P)^{abc} | a, b, c \in Z_q; T \leftarrow A^{O_{DBDH}}(\Gamma, aP, bP, cP)].$$

Here O_{DBDH} denotes a decision bilinear Diffie-Hellman oracle which on input (aP, bP, cP, T) outputs 1 if $T = e(P,P)^{abc}$ and 0 otherwise. q_{DBDH} denotes the maximum number of queries that A asks its decision oracle [38,39]

Definition 3. Given a symmetric bilinear group description Γ, it is said that the computational Diffie-Hellman (*CDH*) assumption holds in G_1 if the advantage of any PPT adversary A as defined below is negligible.

$$Adv_{\Gamma}^{CDH}(A) := Pr[Q = abP | a, b \in Z_q; Q \leftarrow A(\Gamma, aP, bP)].$$

3. Definition and security model of CLAEKS

The entities involved are *KGC*, data senders, data receivers and the cloud server. There are also eight PPT algorithms (*Setup, ExtractPartialPrivateKey, SetSecretValue, SetPrivateKey, SetPublicKey, CLAEKS, TrapdoorGen, Test*). *KGC* generates system parameters together with data sender/receiver's partial private keys using *Setup* and *ExtractPartialPrivateKey*. In addition, the sender/receiver private keys have a second part which is determined by themselves using *SetSecretValue*. Full private keys and public keys are naturally generated by *SetPrivateKey* and *SetPublicKey*. A data sender encrypts keywords in each document using *CLAEKS* and stores the results on the server. In order to search for keywords, data receiver uses *TrapdoorGen* to create the corresponding trapdoor and sends it to the server who runs *Test* to find documents containing the searched keywords. The details of the input/output of these algorithms are provided in Section 4.

We now provide our enhanced security model of a CLAEKS (see Section 1.1). In certificateless cryptography, two types of PPT attackers are considered; a type 1 attacker A_1, who doesn't have access to the master secret key but can replace the public keys of users and a type 2 attacker A_2 who has access to the master secret key but cannot replace any user's public key. We remark that throughout the rest of this paper, by A_1 and A_2, we mean a type 1 and type 2 attacker in CLAEKS, respectively. Now, for a CLAEKS to be secure, it should provide both ciphertext and trapdoor security as defined in the following subsections.

3.1. Ciphertext security

Ciphertext security of a CLAEKS scheme means that it should be (computationally) impossible for an attacker of either type A_1 or A_2 to distinguish between the ciphertexts of the two chosen challenge keywords, while he is allowed to obtain ciphertexts corresponding to any keywords and trapdoors for any non-challenged keywords. Now, depending on its type, the attacker is given access to have responses for a range of queries. For example, type 1 attacker is allowed to launch a request-public-key query while A_2 is permitted to access the master key. Therefore, to describe the ciphertext security for a CLAEKS (*Setup, ExtractPartialPrivateKey, SetSecretValue, SetPrivateKey, SetPublicKey, CLAEKS, TrapdoorGen, Test*), the following two games are defined.

Game I: This game is performed between the challenger $\mathcal{B}$ and A_1.

- *Initialization*. $\mathcal{B}$ runs *Setup* to generate a master secret key (*msk*) and public parameters *prms*. $\mathcal{B}$ keeps *msk* secret and gives *prms* to A_1.
- *Phase 1*. $\mathcal{B}$ responds to (a polynomially bounded number of) the following queries made adaptively by A_1.
  - ReqPK(ID_u). The public key of an identity ID_u is queried. Response: P_{ID_u}.
  - ExtPPK(ID_u). The partial private key of ID_u is queried. Response: D_{ID_u}.

- RepPK(ID_u, P'_{ID_u}). In response, the public key of ID_u is replaced by P'_{ID_u} and his secret value by $\perp$, where $\perp$ is an empty string (this assumption holds throughout the rest of the paper).
- ExtSV(ID_u). The secret value of ID_u is queried. Response: x_{ID_u}. The public key of u should not have been replaced by A_1.
- qCLAEKS(ID_S, ID_R, w). The output of *CLAEKS* on the sender ID_S, receiver ID_R and keyword w is returned.
- qTrapdoor(ID_S, ID_R, w). The output of *TrapdoorGen* on the sender ID_S, receiver ID_R and keyword w is returned.

- *Challenge.* A_1 outputs a sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$, and two challenge keywords w_0 and $w_1 (\neq w_0)$. Now, $\mathcal{B}$ chooses a random bit b and performs *CLAEKS* $(prms, ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, w_b)$ to generate the target searchable ciphertext $CT^\diamond$ and returns it to A_1.
- *Phase 2*. The attacker can continue to probe the challenger as in Phase 1.
- *Guess*. The attacker returns a bit b' and wins the game if $b = b'$, subject to the followings:
  - A_1 has never queried ExtPPK($ID_S^\diamond$) or ExtPPK($ID_R^\diamond$).
  - A_1 has never queried qTrapdoor($ID_S^\diamond, ID_R^\diamond, w_i$) for $i = 0, 1$.

The advantage of A_1 is defined as

$$Adv_{A_1}^{CLAEKS-CT-ind-CKA} = |2Pr[b' = b] - 1|.$$

Game II: This game is performed between the challenger $\mathcal{B}$ and A_2.

- *Initialization*. $\mathcal{B}$ runs *Setup* to generate a master secret key *msk* and the public parameters *prms*. $\mathcal{B}$ gives *msk* and *prms* to A_2.
- *Phase 1.* A_2 may adaptively make a polynomially bounded number of queries as in Game I. The only constraint is that A_2 is not allowed to perform RepPK(.) queries. Note that since A_2 knows the master secret key, he can compute the partial private key of any identity by himself.
- *Challenge.* A_2 outputs a sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$ and two challenge keywords w_0 and $w_1 (\neq w_0)$. Now, $\mathcal{B}$ chooses a random bit b and performs CLAEKS($prms, ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, w_b$) to generate the target searchable ciphertext $CT^\diamond$ and returns it to A_2.
- *Phase 2*. The attacker can continue to probe the challenger as in Phase 1.
- *Guess*. The attacker returns a bit b' and wins the game if $b = b'$, subject to the followings:
  - A_2 has never queried ExtSV($ID_S^\diamond$) or ExtSV($ID_R^\diamond$).
  - A_2 has never queried qTrapdoor($ID_S^\diamond, ID_R^\diamond, w_i$) for $i = 0, 1$.

The advantage of A_{II} is defined as

$$Adv_{A_2}^{CLAEKS-CT-ind-CKA} = |2Pr[b' = b] - 1|.$$

Definition 4. We say that a CLAEKS scheme provides ciphertext indistinguishability against an adaptive chosen keyword attack if for any polynomial-time attackers $A_i (i \in \{1, 2\})$, $Adv_{A_i}^{CLAEKS-CT-ind-CKA}$ is negligible.

3.2. Trapdoor security

Trapdoor security of a CLAEKS scheme means that it should be (computationally) impossible for an attacker of either type A_1 or A_2 to distinguish between the trapdoors of the two chosen challenge keywords, while he is allowed to obtain ciphertexts corresponding to any keywords and trapdoors for any non-challenged keywords. As before, trapdoor security for a CLAEKS (*Setup, Extract-PartialPrivateKey, SetSecretValue, SetPrivateKey, SetPublicKey, CLAEKS, TrapdoorGen, Test*) is defined by the following two games.

Game III: This game is performed between the challenger $\mathcal{B}$ and A_1.

- *Initialization*. $\mathcal{B}$ runs *Setup* to generate a master secret key *msk* and the public parameters *prms*. $\mathcal{B}$ keeps *msk* secret and gives *prms* to A_1.
- *Phase 1.* A_1 may adaptively make a polynomially bounded number of queries as in Game I and $\mathcal{B}$ answers to them in the same way.
- *Challenge.* A_1 outputs a sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$, and two challenge keywords w_0 and $w_1 (\neq w_0)$. Now, $\mathcal{B}$ chooses a bit $b \in \{0, 1\}$ uniformly at random and performs *TrapdoorGen* $(prms, ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, w_b)$ to generate the target trapdoor $T^\diamond$ and returns it to A_1.
- *Phase 2*. The attacker can continue to probe the challenger as in Phase 1.
- *Guess*. The attacker returns a bit b' and wins the game if $b = b'$, subject to the following conditions:
  - A_1 has never queried the partial private key corresponding to $ID_S^\diamond$ or $ID_R^\diamond$.
  - A_1 has never queried *qCLAEKS* $(ID_S^\diamond, ID_R^\diamond, w_i)$ for $i = 0, 1$.
  - A_1 has never queried qTrapdoor($ID_S^\diamond, ID_R^\diamond, w_i$) where $i = 0, 1$.

The advantage of A_1 is defined as follows:

$$Adv_{A_1}^{CLAEKS-Trapdoor-ind-CKA} = |2Pr[b' = b] - 1|.$$

Game IV: This game is performed between the challenger $\mathcal{B}$ and A_2.

- *Initialization*. $\mathcal{B}$ runs *Setup* to generate a master secret key *msk* and the public parameters *prms*. $\mathcal{B}$ gives *msk* and *prms* to A_2.
- *Phase 1.* A_2 may adaptively make a polynomially bounded number of queries as in Game II and $\mathcal{B}$ answers to them in the same way.
- *Challenge.* A_2 outputs a data sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$ and two challenge keywords w_0 and $w_1 (\neq w_0)$. Now, $\mathcal{B}$ chooses a bit $b \in \{0, 1\}$ uniformly at random and performs $TrapdoorGen(prms, ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, w_b)$ to generate the target trapdoor $T^\diamond$ and returns it to A_2.
- *Phase 2*. The adversary can continue to probe the challenger as in Phase 1.
- *Guess*. The adversary returns a bit b' and wins the game if $b = b'$, subject to the following conditions:
  - A_2 has never queried the secret value corresponding to $ID_S^\diamond$ or $ID_R^\diamond$.
  - A_2 has never queried *qCLAEKS* $(ID_S^\diamond, ID_R^\diamond, w_i)$ for $i = 0, 1$.
  - A_2 has never queried qTrapdoor($ID_S^\diamond, ID_R^\diamond, w_i$) where $i = 0, 1$.

As in Game III, the advantage of A_2 is defined as follows:

$$Adv_{A_2}^{CLAEKS-Trapdoor-ind-CKA} = |2Pr[b' = b] - 1|.$$

Definition 5. We say that a CLAEKS scheme provides trapdoor indistinguishability against an adaptive chosen keyword attack if for any polynomial-time attackers A_i $(i = 1, 2)$, $Adv_{A_i}^{CLAEKS-Trapdoor-ind-CKA}$ is negligible.

4. The proposed CLAEKS scheme

In this section, the details of the proposed CLAEKS scheme are provided. The proposed scheme consists of the following eight PPT algorithms:

- **Setup:** performed by *KGC* with the security parameter λ as input.
  - *Process:*
    1. Choose two cyclic groups G_1 and G_2 of prime order $q > 2^\lambda$ and a bilinear map e: $G_1 \times G_1 \rightarrow G_2$.

2. Choose $s \in Z_q^*$ randomly as the master secret key and a generator $P \in G_1$, and compute $P_{Pub} = sP \in G_1$ as the master public key.
3. Choose three cryptographic hash functions H: $\{0,1\}^* \rightarrow G_1$, $h_1 : \{0,1\}^* \times G_1 \times \{0,1\}^* \times G_1 \times G_2 \times G_1 \times \{0,1\}^* \rightarrow Z_q^*$, and $h_2 : Z_q^* \times G_1 \rightarrow Z_q^*$.

– *Output:* The master secret key s which will be secured by *KGC* and the system parameters $prms = (\lambda, G_1, G_2, e, q, P, P_{Pub}, H, h_1, h_2)$ which will be published.

ExtractPartialPrivateKey: performed by *KGC*.
– Input: *prms*, master secret key s and a user's identity $ID_u \in \{0,1\}^*$.
– Process: Compute $D_{ID_u} = sH(ID_u)$.
– Output: Partial private key D_{ID_u} which will be sent securely to the user with identity ID_u.

SetSecretValue: performed by each user of the system.
– Input: *prms* and user's identity ID_u.
– Process: Select a random value $x_{ID_u} \in Z_q^*$ as the user's secret value.
– Output: x_{ID_u} which will be secured by the user.

SetPrivateKey: performed by each user of the system.
– Input: The user's secret value x_{ID_u} and his partial private key (D_{ID_u}).
– Process: Set $SK_{ID_u} = (x_{ID_u}, D_{ID_u})$ as the user's private key.
– Output: SK_{ID_u} which will be secured by the user.

SetPublicKey: performed by each user of the system.
– Input: *prms* and the user's secret value x_{ID_u}.
– Process: Compute $P_{ID_u} = x_{ID_u}P$.
– Output: P_{ID_u} which will be published.

CLAEKS: performed by the data sender.
– Input: *prms*, the data sender's identity ID_S, his public key P_{ID_S} and his private key $SK_{ID_S} = (x_{ID_S}, D_{ID_S})$, the receiver's identity ID_R and his public key P_{ID_R}, and a keyword w.
– Process:
  • Choose a random number $r \in Z_q^*$.
  • Compute

$$K_1 = e(D_{ID_S}, H(ID_R)),$$
$$K_2 = x_{ID_S}P_{ID_R},$$
$$C_1 = rP,$$
$$C_2 = rh_2(h_1(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w), C_1)P.$$

– Output: The searchable ciphertext $CT = (C_1, C_2)$ which will be uploaded to the server.

TrapdoorGen: performed by the receiver.
– Input: *prms*, the data sender's identity ID_S and his public key P_{ID_S}, the receiver's identity ID_R, his public key P_{ID_R}, his private key $SK_{ID_R} = (x_{ID_R}, d_{ID_R})$ and a keyword w.
– Process:
  • Compute

$$K_1 = e(H(ID_S), D_{ID_R}),$$
$$K_2 = x_{ID_R}P_{ID_S},$$
$$T_w = h_1(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w).$$

– Output: T_w which will be sent to the server.

Test: performed by the server.
– Input: *prms*, trapdoor T_w and ciphertext $CT = (C_1, C_2)$.
– Process:
  • Check if the following equation holds:

$$C_2 = h_2(T_w, C_1)C_1.$$

– Output: 1 if the above equation holds and 0 otherwise.

5. Security analysis

In this section, we analyze the security of the proposed scheme which is denoted throughout this section by ($Setup_\Pi$, $ExtractPartialPrivateKey_\Pi$, $SetSecretValue_\Pi$, $SetPrivateKey_\Pi$, $SetPublicKey_\Pi$, $CLAEKS_\Pi$, $TrapdoorGen_\Pi$, $Test_\Pi$). Theorems 1 and 2 show that the proposed CLAEKS scheme provides the required ciphertext security and trapdoor security, respectively. As a consequence of Theorem 2 of this paper and Theorem 5 of [6], it can be concluded that the proposed scheme is secure against KGA. Note that in the proposed scheme, the inside attacker (i.e., the server) has no advantage compared to the outside attacker and therefore, the proposed scheme is secure against KGAs performed by both the inside and outside attackers.

Theorem 1. *Under GBDH and CDH assumptions, the proposed CLAEKS scheme provides ciphertext indistinguishability against an adaptive chosen keyword attack.*

This theorem will be proved through Lemmas 1 and 2.

Lemma 1. *Let A_1 be a type 1 attacker of certificateless cryptography. Suppose that A_1 is able to break the proposed CLAEKS scheme during Game I with non-negligible advantage ϵ. Then, using A_1, a PPT algorithm $\mathcal{B}$ can be constructed who can solve GBDH problem with the advantage $\epsilon' \geq \frac{2\epsilon}{q_H(q_H+1)}$.*

Proof. Let (P, aP, bP, cP) be an instance of GBDH problem, where $a, b, c \in Z_q^*$ are random choices. In the following, we show how $\mathcal{B}$ can use A_1 to solve this instance of GBDH problem.

- *Initialization.* $\mathcal{B}$ runs $Setup_\Pi$, except that it sets $P_{Pub} = aP$. It returns *prms* to A_1 and chooses $l_1, l_2 \leq q_H$ randomly as the indices of the challenge identities for the data sender and the data receiver, respectively. Here q_H is the maximum number of queries that could be queried the H oracle for different identities.
- *Phase 1.* To solve the GBDH problem, $\mathcal{B}$ replies to queries made by A_1 as described in the following. Note that to avoid collision and consistently respond to these queries, $\mathcal{B}$ has to maintain some lists, which are all initially empty (this assumption holds throughout the rest of the paper).
  H queries: On the ith (non-repeated) query (ID_i), $\mathcal{B}$ searches L for the entry $(i, ID_i, \mu_{ID_i}, H(ID_i))$. If no such entry is found,
  – if $i \notin \{l_1, l_2\}$, $\mathcal{B}$ chooses $\mu_{ID_i} \in Z_q^*$ randomly, sets $H(ID_i) = \mu_{ID_i}P$, adds $(i, ID_i, \mu_{ID_i}, H(ID_i))$ to L and returns $H(ID_i)$.
  – if $i = l_1$, $\mathcal{B}$ sets $H(ID_i) = bP$, adds $(l_1, ID_i, \perp, H(ID_i))$ to L and returns $H(ID_i)$.
  – if $i = l_2$, $\mathcal{B}$ sets $H(ID_i) = cP$, adds $(l_2, ID_i, \perp, H(ID_i))$ to L and returns $H(ID_i)$.
  h_1 *queries:* On input $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w)$, $\mathcal{B}$
  – searches L_1 for the entry $((ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w), h)$. If no such entry is found, it
    1. retrieves $(i, ID_S, \mu_{ID_S}, H(ID_S))$ and $(j, ID_R, \mu_{ID_R}, H(ID_R))$ by calling $H(ID_S)$ and $H(ID_R)$.
    2. checks if the output of DBDH oracle on the tuple $(H(ID_S), H(ID_R), P_{Pub}, K_1)$ is 1 and $e(P_{ID_S}, P_{ID_R}) = e(K_2, P)$.
    3. if $\{i, j\} = \{l_1, l_2\}$, returns K_1 as the answer to the instance of GBDHP and stops.
    4. chooses $h \in Z_q^*$ randomly and inserts $((ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w), h)$ to L_1.
  – returns h.
  h_2 *queries:* On input (h, C_1), $\mathcal{B}$ searches L_3 for the entry $((h, C_1), h_2(h, C_1))$. If no such entry is found, it chooses $h_2(h, C_1) \in Z_q^*$ randomly and inserts $((h, C_1), h_2(h, C_1))$ to L_3. returns $h_2(h, C_1)$.
  CreateUser(ID): On input ID, $\mathcal{B}$ goes through the list L_k for a tuple $(ID, *, *, *)$. If no such a tuple is found, it calls $H(ID)$ and obtains $(i, ID, \mu_{ID}, H(ID))$.

– If $i \notin \{l_1, l_2\}$, it chooses random values $x_{ID} \in Z_q^*$, and computes $P_{ID} = x_{ID}P$. It computes $D_{ID} = \mu_{ID}P_{Pub}$ and adds (ID, D_{ID}, x_{ID}, P_{ID}) to L_k.
– If $i \in \{l_1, l_2\}$, it chooses $x_{ID} \in Z_q^*$, computes $P_{ID} = x_{ID}P$ and adds (ID, $\perp$, x_{ID}, P_{ID}) to L_k.

ExtPPK(ID): On input ID, $\mathcal{B}$ first invokes CreateUser(ID) and obtains (ID, D_{ID}, x_{ID}, P_{ID}). It returns D_{ID} to A_1 if $D_{ID} \neq \perp$ and aborts otherwise.

ExtSV(ID): On input ID, $\mathcal{B}$ obtains (ID, D_{ID}, x_{ID}, P_{ID}) by invoking CreateUser(ID) and returns x_{ID}. Note that if the corresponding public key to ID is replaced by A_1, then the answer to this query would be $\perp$.

ReqPK queries: On input ID, $\mathcal{B}$ obtains (ID, D_{ID}, x_{ID}, P_{ID}) by calling CreateUser(ID) and returns P_{ID}.

RepPK queries: On input (ID, P'_{ID}), $\mathcal{B}$ obtains (ID, D_{ID}, x_{ID}, P_{ID}) by calling CreateUser(ID). Then, it updates the list L_K by replacing (ID, D_{ID}, x_{ID}, P_{ID}) with $(ID, D_{ID}, \perp, P'_{ID})$.

qCLAEKS(ID_S, ID_R, w): On input (ID_S, ID_R, w), $\mathcal{B}$
– retrieves $(i, ID_S, \mu_{ID_S}, H(ID_S))$ and $(j, ID_R, \mu_{ID_R}, H(ID_R))$ from L and $(ID_S, D_{ID_S}, x_{ID_S}, P_{ID_S})$ and $(ID_R, D_{ID_R}, x_{ID_R}, P_{ID_R})$ from L_k.
– selects $r \in Z_q^*$ randomly.
– if $\{i, j\} = \{l_1, l_2\}$:
  1. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, *, *, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, \perp, \perp, w, h)$ to L_1. In both cases, $\mathcal{B}$ retrieves $((h, C_1), h_2(h, C_1))$ by calling $h_2(h, C_1)$ and computes
  $$C_1 = rP \text{ and } C_2 = rh_2(h, C_1)P.$$
  2. returns (C_1, C_2).
– Otherwise, (i.e., in case $i \notin \{l_1, l_2\}$ or $j \notin \{l_1, l_2\}$ which without loss of generality, we assume that $i \notin \{l_1, l_2\}$):
  1. computes $K_1 = e(\mu_{ID_S}P_{Pub}, H(ID_R))$.
  2. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, *, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, \perp, w, h)$ to L_1. In both cases, $\mathcal{B}$ retrieves $((h, C_1), h_2(h, C_1))$ by calling $h_2(h, C_1)$ and computes
  $$C_1 = rP \text{ and } C_2 = rh_2(h, C_1)P.$$
  3. returns (C_1, C_2).

qTrapdoor(ID_S, ID_R, w): On input (ID_S, ID_R, w), $\mathcal{B}$
– retrieves $(i, ID_S, \mu_{ID_S}, H(ID_S))$ and $(j, ID_R, \mu_{ID_R}, H(ID_R))$ from L and $(ID_S, D_{ID_S}, x_{ID_S}, P_{ID_S})$ and $(ID_R, D_{ID_R}, x_{ID_R}, P_{ID_R})$ from L_k.
– selects $r \in Z_q^*$ randomly.
– if $\{i, j\} = \{l_1, l_2\}$:
  1. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, *, *, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, \perp, \perp, w, h)$ to L_1. In both cases, sets $T_w = h$.
– Otherwise, (i.e., in case $i \notin \{l_1, l_2\}$ or $j \notin \{l_1, l_2\}$ which without loss of generality, we assume that $i \notin \{l_1, l_2\}$):
  1. computes $K_1 = e(\mu_{ID_S}P_{Pub}, H(ID_R))$.
  2. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, *, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, \perp, w, h)$ to L_1. In both cases, sets $T_w = h$.
– returns T_w.

- *Challenge.* Eventually, A_1 outputs a data sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$ and two challenge keywords w_0 and $w_1 (\neq w_0)$. To generate the challenge ciphertext, $\mathcal{B}$
  1. calls $H(ID_S^\diamond)$ and $H(ID_R^\diamond)$ to obtain $(i, ID_S^\diamond, \mu_{ID_S^\diamond}, H(ID_S^\diamond))$ and $(j, ID_R^\diamond, \mu_{ID_R^\diamond}, H(ID_R^\diamond))$, and aborts if $\{i, j\} \neq \{l_1, l_2\}$.
  2. obtains $(ID_S^\diamond, D_{ID_S^\diamond}, x_{ID_S^\diamond}, P_{ID_S^\diamond})$ and $(ID_R^\diamond, D_{ID_R^\diamond}, x_{ID_R^\diamond}, P_{ID_R^\diamond})$ by calling CreateUser($ID_S^\diamond$) and CreateUser($ID_R^\diamond$), respectively.
  3. chooses $b \in \{0, 1\}$ and $r' \in Z_q$ randomly.
  4. searches L_1 for the tuple $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, *, *, w_b, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, \perp, \perp, w_b, h)$ to L_1. In both cases, $\mathcal{B}$ retrieves $((h, C_1), h_2(h, C_1))$ by calling $h_2(h, C_1)$ and computes
  $$C_1 = rP \text{ and } C_2 = rh_2(h, C_1)P.$$
  5. returns (C_1, C_2).
- *Phase 2.* Queries made by A_1 during Phase 2 are treated as in Phase 1.
- *Guess.* Finally, A_1 will output $b' \in \{0, 1\}$ as its guess.

Algorithm $\mathcal{B}$ exactly follows the proposed scheme except that it simulates the hash functions with random oracles. Therefore, our simulation is perfect. Now, since 1) l_1 and l_2 are independent of adversary's view, and 2) L contains at most q_H elements, with probability $\frac{2}{q_H(q_H+1)}$ the adversary will output the identities ID_{l_1} and ID_{l_2} as the challenge identities for the data sender and the receiver, respectively. In this case, given that h_1 is modeled as a random oracle, the adversary's advantage would be negligible unless $(ID_{l_1}, P_{l_1}, ID_{l_2}, P_{l_2}, K_1^\diamond, K_2^\diamond, w_b, h)$ appears on L_3. If this tuple appears on L_3, $\mathcal{B}$ certainly is able to solve the *GBDH* problem. Following this observation and the fact that $\mathcal{B}$ has made at most q_{h_1} calls to O_{DBDH}, it can be concluded that $Adv_\Gamma^{GBDH}(\mathcal{B}, q_{h_1}) \geq \frac{2\epsilon}{q_H(q_H+1)}$ (the same as ϵ) is non-negligible. □

Lemma 2. *Let A_2 be a type 2 attacker of certificateless cryptography. Suppose that A_2 is able to break the proposed CLAEKS scheme during Game II with non-negligible advantage ϵ. Then, using A_2, a PPT algorithm $\mathcal{B}$ can be constructed who can solve the CDH problem with the advantage $\epsilon' \geq \frac{2\epsilon}{q_H(q_H+1)}$.*

Proof. Let (P, aP, bP) be an instance of CDH problem, where $a, b \in Z_q^*$ are random choices. In the following, we show how $\mathcal{B}$ can use A_2 to solve this instance of the CDH problem.

- *Initialization.* $\mathcal{B}$ runs $Setup_\Pi$. It returns *prms* and msk to A_2 and chooses $l_1, l_2 \leq q_H$ randomly as the indices of the challenge identities for the data sender and the receiver where q_H is the maximum number of queries that could be queried the H oracle for different identities.
- *Phase 1.* To solve the CDH problem, $\mathcal{B}$ replies to queries made by A_2 as in the following.

  H queries: On the ith (non-repeated) query (ID_i), $\mathcal{B}$ searches L for the entry (i, ID_i, $H(ID_i)$). If no such entry is found,
  – $\mathcal{B}$ chooses $H(ID_i) \in G_1$ randomly, adds (i, ID_i, $H(ID_i)$) to L and returns $H(ID_i)$.

  h_1 *queries:* On input $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w)$, $\mathcal{B}$
  – searches L_1 for the entry $((ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w), h)$. If no such entry is found, it
    1. retrieves (i, ID_S, $H(ID_S)$) and (j, ID_R, $H(ID_R)$) by calling $H(ID_S)$ and $H(ID_R)$.
    2. checks if $e(sH(ID_S), H(ID_R)) = K_1$ and $e(P_{ID_S}, P_{ID_R}) = e(K_2, P)$.
    3. if $\{i, j\} = \{l_1, l_2\}$, returns K_2 as the answer to the instance of CDHP and stops.
    4. chooses $h \in Z_q^*$ randomly and inserts $((ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w), h)$ to L_1.
  – returns h.

  h_2 *queries:* On input (h, C_1), $\mathcal{B}$ searches L_3 for the entry $((h, C_1), h_2(h, C_1))$. If no such entry is found, it chooses $h_2(h, C_1) \in Z_q^*$ randomly and inserts $((h, C_1), h_2(h, C_1))$ to L_3. In both cases, it returns $h_2(h, C_1)$.

  CreateUser(ID): On input ID, $\mathcal{B}$ goes through the list L_k for a tuple (ID, *, *, *). If no such a tuple is found,
  – It calls $H(ID)$ and obtains (i, ID, $H(ID)$).

- If $i \notin \{l_1, l_2\}$, it chooses a random value $x_{ID} \in Z_q^*$, and computes $P_{ID} = x_{ID}P$. It computes $D_{ID} = sH(ID)$ and adds (ID, D_{ID}, x_{ID}, P_{ID}) to L_k.
- If $i = l_1$, it sets $P_{ID} = aP$, computes $D_{ID} = sH(ID)$ and adds (ID, D_{ID}, $\perp$, P_{ID}) to L_k.
- If $i = l_2$, it sets $P_{ID} = bP$, computes $D_{ID} = sH(ID)$ and adds (ID, D_{ID}, $\perp$, P_{ID}) to L_k.

ExtSV(ID): On input ID, $\mathcal{B}$ obtains (ID, D_{ID}, x_{ID}, P_{ID}) by calling CreateUser(ID) and returns x_{ID}. It returns x_{ID} to A_1 if $x_{ID} \neq \perp$ and aborts otherwise.

ReqPK queries: On input ID, $\mathcal{B}$ obtains (ID, D_{ID}, x_{ID}, P_{ID}) by calling CreateUser(ID) and returns P_{ID}.

qCLAEKS(ID_S, ID_R, w): On input (ID_S, ID_R, w), $\mathcal{B}$ performs the following steps:

- retrieves (i, ID_S, $H(ID_S)$) and (j, ID_R, $H(ID_R)$) from L and ($ID_S, D_{ID_S}, x_{ID_S}, P_{ID_S}$) and ($ID_R, D_{ID_R}, x_{ID_R}, P_{ID_R}$) from L_k.
- selects $r \in Z_q^*$ randomly.
- computes $K_1 = e(sH(ID_S), H(ID_R))$.
- if $\{i, j\} = \{l_1, l_2\}$:
  1. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, *, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, \perp, w, h)$ to L_1. In both cases, using h, $\mathcal{B}$ retrieves $((h, C_1), h_2(h, C_1))$ by calling $h_2(h, C_1)$ and computes
  $$C_1 = rP \text{ and } C_2 = rh_2(h, C_1)P.$$
  2. returns (C_1, C_2).
- Otherwise, (i.e., in case $i \notin \{l_1, l_2\}$ or $j \notin \{l_1, l_2\}$ which without loss of generality, we assume that $i \notin \{l_1, l_2\}$):
  1. computes $K_2 = x_{ID_S} P_{ID_R}$.
  2. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w, h)$ to L_1. In both cases, using h, $\mathcal{B}$ retrieves $((h, C_1), h_2(h, C_1))$ by calling $h_2(h, C_1)$ and computes
  $$C_1 = rP \text{ and } C_2 = rh_2(h, C_1)P.$$
  3. returns (C_1, C_2).

qTrapdoor(ID_S, ID_R, w): On input (ID_S, ID_R, w), $\mathcal{B}$

- retrieves (i, ID_S, $H(ID_S)$) and (j, ID_R, $H(ID_R)$) from L and ($ID_S, D_{ID_S}, x_{ID_S}, P_{ID_S}$) and ($ID_R, D_{ID_R}, x_{ID_R}, P_{ID_R}$) from L_k.
- computes $K_1 = e(sH(ID_S), H(ID_R))$.
- if $\{i, j\} = \{l_1, l_2\}$:
  1. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, *, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, \perp, w, h)$ to L_1. In both cases, sets $T_w = h$.
- Otherwise, (i.e., in case $i \notin \{l_1, l_2\}$ or $j \notin \{l_1, l_2\}$ which without loss of generality, we assume that $i \notin \{l_1, l_2\}$):
  1. computes $K_2 = x_{ID_S} P_{ID_R}$.
  2. searches L_1 for the tuple $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S, P_{ID_S}, ID_R, P_{ID_R}, K_1, K_2, w, h)$ to L_1. In both cases, sets $T_w = h$.
- returns T_w.

- *Challenge.* Eventually, A_2 outputs a data sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$ and two challenge keywords w_0 and $w_1 (\neq w_0)$. To generate the challenge ciphertext, $\mathcal{B}$
  1. calls $H(ID_S^\diamond)$ and $H(ID_R^\diamond)$ to obtain $(i, ID_S^\diamond, H(ID_S^\diamond))$ and $(j, ID_R^\diamond, H(ID_R^\diamond))$, and aborts if $\{i, j\} \neq \{l_1, l_2\}$.
  2. obtains $(ID_S^\diamond, D_{ID_S^\diamond}, x_{ID_S^\diamond}, P_{ID_S^\diamond})$ and $(ID_R^\diamond, D_{ID_R^\diamond}, x_{ID_R^\diamond}, P_{ID_R^\diamond})$ by calling CreateUser($ID_S^\diamond$) and CreateUser($ID_R^\diamond$), respectively.
  3. chooses $b \in \{0, 1\}$ and $r \in Z_q^*$ randomly.
  4. computes $K_1 = e(sH(ID_S), H(ID_R))$.
  5. searches L_1 for the tuple $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, K_1, *, w_b, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, K_1, \perp, w_b, h)$ to L_1. In both cases, using h, $\mathcal{B}$ retrieves $((h, C_1), h_2(h, C_1))$ by calling $h_2(h, C_1)$ and computes
  $$C_1 = rP \text{ and } C_2 = rh_2(h, C_1)P.$$
  6. returns (C_1, C_2).
- *Phase 2.* Queries made by A_2 during Phase 2 are treated as in Phase 1.
- *Guess.* Finally, A_2 will output $b' \in \{0, 1\}$ as its guess.

Note that, in this lemma, the security of the scheme is reduced to CDH problem (it was GBDH in the previous lemma and there is no need for DBDH oracle). Therefore, with similar arguments as in previous lemma, it can be concluded that $Adv_\Gamma^{CDH}(\mathcal{B}) \geq \frac{2\epsilon}{q_H(q_H+1)}$ which (the same as ϵ) is non-negligible. □

Theorem 2. *Under GBDH and CDH assumptions, the proposed CLAEKS scheme provides trapdoor indistinguishability against an adaptive chosen keyword attack.*

This theorem will be proved through Lemmas 3 and 4.

Lemma 3. *Let A_1 be a type 1 attacker of certificateless cryptography. Suppose that A_1 is able to break the proposed CLAEKS scheme during Game III with non-negligible advantage ϵ. Then, using A_1, a PPT algorithm $\mathcal{B}$ can be constructed who can solve the GBDH problem with the advantage $\epsilon' \geq \frac{2\epsilon}{q_H(q_H+1)}$.*

Proof. Let (P, aP, bP, cP) be an instance of GBDH problem, where $a, b, c \in Z_q^*$ are random choices. In the following, we show how $\mathcal{B}$ can use A_1 to solve this instance of the GBDH problem.

- *Initialization.* $\mathcal{B}$ runs $Setup_\Pi$, except that it sets $P_{Pub} = aP$. It returns $prms$ to A_1 and chooses l_1, $l_2 \leq q_H$ randomly as the indices of the challenge identities for the data sender and the receiver, respectively where q_H is the maximum number of queries that could be queried the H oracle for different identities.
- *Phase 1.* To solve the GBDH problem, $\mathcal{B}$ replies to queries made by A_1 in the same way it did in Game 1.
- *Challenge.* Eventually, A_1 outputs a data sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$ and two challenge keywords w_0 and $w_1 (\neq w_0)$. To generate the challenge trapdoor, $\mathcal{B}$:
  1. calls $H(ID_S^\diamond)$ and $H(ID_R^\diamond)$ to obtain $(i, ID_S^\diamond, \mu_{ID_S^\diamond}, H(ID_S^\diamond))$ and $(j, ID_R^\diamond, \mu_{ID_R^\diamond}, H(ID_R^\diamond))$, and aborts if $\{i, j\} \neq \{l_1, l_2\}$.
  2. obtains $(ID_S^\diamond, D_{ID_S^\diamond}, x_{ID_S^\diamond}, P_{ID_S^\diamond})$ and $(ID_R^\diamond, D_{ID_R^\diamond}, x_{ID_R^\diamond}, P_{ID_R^\diamond})$ by calling CreateUser($ID_S^\diamond$) and CreateUser($ID_R^\diamond$), respectively.
  3. chooses $b \in \{0, 1\}$ randomly.
  4. searches L_1 for the tuple $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, *, *, w_b, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, \perp, \perp, w_b, h)$ to L_1. In both cases, sets $T_{w_b} = h$.
  5. returns T_{w_b} as the challenge trapdoor.
- *Phase 2.* Queries made by A_1 during Phase 2 are treated as in Phase 1.
- *Guess.* Finally, A_1 will output $b' \in \{0, 1\}$ as its guess.

With similar arguments as in Lemma 1, it can be concluded that $Adv_\Gamma^{GBDH}(\mathcal{B}, q_{h_1}) \geq \frac{2\epsilon}{q_H(q_H+1)}$ which (the same as ϵ) is non-negligible. □

Lemma 4. *Let A_2 be a type 2 attacker of certificateless cryptography. Suppose that A_2 is able to break the proposed CLAEKS scheme during Game IV with non-negligible advantage ϵ. Then, using A_2, a PPT algorithm $\mathcal{B}$ can be constructed who can solve the CDH problem with the advantage $\epsilon' \geq \frac{2\epsilon}{q_H(q_H+1)}$.*

Table 1
Security comparison.

	[27]	[30]	[31]	[32]	[33]	[34]	[35]	Ours
Ciphertext indistinguishability	No	Yes	No	Yes	No	No	No	Yes
Security against outside KGA	No	No	No	Yes	Yes	No	Yes	Yes
Security against inside KGA	No	No	No	No	No	No	Yes	Yes

Table 2
Computation cost (ms).

	Ciphertext generation	Trapdoor generation	Test
[27]	$3T_H + 4T_{sm} + 5T_p = 14.676$	$T_H + 3T_{sm} = 1.267$	$2T_{sm} + T_p = 3.108$
[30]	$T_H + 4T_{sm} + T_p = 4.064$	$T_H + T_{sm} = 0.645$	$T_p = 2.486$
[31]	$3T_H + 4T_{sm} + 3T_p = 9.704$	$T_H + T_{sm} = 0.645$	$2T_H + T_{sm} + T_p = 3.465$
[33]	$T_H + 5T_{sm} = 1.889$	$T_H + 3T_{sm} + T_p = 3.753$	$2T_{sm} + 2T_p = 5.594$
[34]	$T_H + 5T_{sm} = 1.889$	$T_H + 7T_{sm} + T_p = 4.997$	$4T_{sm} + 2T_p = 6.216$
[32]	$4T_H + 5T_{sm} + 3T_p = 10.349$	$3T_H + 6T_{sm} + T_p = 5.354$	$3T_{sm} + 4T_p = 10.877$
[35]	$T_H + 5T_{sm} = 1.889$	$T_H + 3T_{sm} + T_p = 3.753$	$2T_{sm} + 2T_p = 5.594$
Ours	$T_H + 3T_{sm} + T_p = 3.753$	$T_H + T_{sm} + T_p = 3.131$	$T_{sm} = 0.311$

Proof. Let (P, aP, bP) be an instance of CDH problem, where $a, b \in Z_q^*$ are random choices. In the following, we show how $\mathcal{B}$ can use A_2 to solve this instance of the CDH problem.

- *Initialization.* $\mathcal{B}$ runs $Setup_\Pi$ algorithm. It returns *prms* and *msk* to A_2 and chooses l_1, $l_2 \leq q_H$ randomly as the indices of the challenge identities for the data sender and the receiver. Here q_H is the maximum number of queries that could be queried the H oracle for different identities.
- *Phase 1.* To solve the CDH problem, $\mathcal{B}$ replies to queries made by A_2. The answer to this queries are the same as those in Game II.
- *Challenge.* Eventually, A_2 outputs a data sender's identity $ID_S^\diamond$, a receiver's identity $ID_R^\diamond$ and two challenge keywords w_0 and $w_1 (\neq w_0)$. To generate the challenge ciphertext, $\mathcal{B}$
  1. carries $H(ID_S^\diamond)$ and $H(ID_R^\diamond)$ to obtain $(i, ID_S^\diamond, H(ID_S^\diamond))$ and $(j, ID_R^\diamond, H(ID_R^\diamond))$, and aborts if $\{i, j\} \neq \{l_1, l_2\}$.
  2. obtains $(ID_S^\diamond, D_{ID_S^\diamond}, x_{ID_S^\diamond}, P_{ID_S^\diamond})$ and $(ID_R^\diamond, D_{ID_R^\diamond}, x_{ID_R^\diamond}, P_{ID_R^\diamond})$ by calling CreateUser($ID_S^\diamond$) and CreateUser($ID_R^\diamond$), respectively.
  3. chooses $b \in \{0, 1\}$ randomly.
  4. computes $K_1 = e(sH(ID_S), H(ID_R))$.
  5. searches L_1 for the tuple $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, K_1, *, w_b, h)$. If no such entry is found, chooses $h \in_R Z_q^*$ and adds $(ID_S^\diamond, P_{ID_S^\diamond}, ID_R^\diamond, P_{ID_R^\diamond}, K_1, \perp, w_b, h)$ to L_1. In both cases, uses h and sets $T_{w_b} = h$.
  6. returns T_{w_b}.
- *Phase 2.* Queries made by A_2 during Phase 2 are treated as in Phase 1.
- *Guess.* Finally, A_2 will output $b' \in \{0, 1\}$ as its guess.

With similar arguments as in Lemma 2, it can be concluded that $Adv_\Gamma^{CDH}(\mathcal{B}) \geq \frac{2\epsilon}{q_H(q_H+1)}$ which (the same as ϵ) is non-negligible. □

6. Performance analysis

To the best of our knowledge, the schemes proposed in [27,30–35] are the only existing searchable encryption schemes in certificateless setting. In this section, we compare the proposed scheme with them first in terms of the provided security requirements and then in terms of computational and communication complexities.

The results of comparison between the proposed scheme and other schemes in terms of the provided security requirements are provided in Table 1. As it can be seen, considering the enhanced security model, the proposed scheme is the only scheme that provides all the needed security requirements.

Table 3
Communication cost comparison.

	Ciphertext size	Trapdoor size
[27]	$\|G_1\| + h = 678$	$3\|G_1\| = 1536$
[30]	$\|G_1\| + h = 678$	$\|G_1\| = 512$
[31]	$\|G_1\| + h = 678$	$\|G_1\| = 512$
[33]	$2\|G_1\| = 1024$	$\|G_2\| = 1024$
[34]	$2\|G_1\| = 1024$	$2\|G_1\| + \|G_2\| = 2048$
[32]	$2\|G_1\| + h = 1184$	$3\|G_1\| = 1536$
[35]	$2\|G_1\| = 1024$	$\|G_2\| = 1024$
Ours	$2\|G_1\| = 1024$	$h = 160$

We now consider computational costs in Table 2 with the following notations:

T_{sm}: The running time of a scalar multiplication operation.
T_H: The running time of a Hash-to-point operation.
T_p: The running time of a bilinear pairing operation.

For the sake of simplicity, we only consider time-consuming operations. To further make the comparisons possible, we use the explicit running times reported for these operations in [40]: $T_H = 0.334$ *ms*, $T_{sm} = 0.311$ *ms*, $T_p = 2.486$ *ms* where *ms* means millisecond. These results are obtained by using PBC (pairing-based cryptography) library [41] running on a laptop running Ubuntu 14.04 LTS with Intel(R) Core™ i3-2310M CPU@2.10 GHz and 4 GB RAM memory. As it can be seen from Table 2 and Fig. 2, the *Test* algorithm of the proposed scheme is very efficient and only the ciphertext generation algorithm of the scheme of [33] is more efficient than ours. Compared to other existing insecure schemes, the trapdoor generation algorithm of the proposed scheme is less efficient. This is the cost paid to obtain the desired security.

Finally, in Table 3 and Fig. 3, we compare the proposed scheme with the related ones in terms of the size of the ciphertext and the trapdoor in each scheme using the following notations:

h: the bit length size of the output of an ordinary hash function,
$|G_i|$: the bit length size of each element of G_i for $i = 1, 2$.

To further simplify the results, we use the bit length sizes reported in [40], i.e., $h = 160$, $|G_1| = 512$ and $|G_2| = 1024$. The results indicate that while the proposed scheme provides all the required security, the size of its trapdoors is less that one-third of the trapdoor-size in the best other existing (insecure) scheme and the size of its ciphertexts is roughly 1.5 times the best existing size.

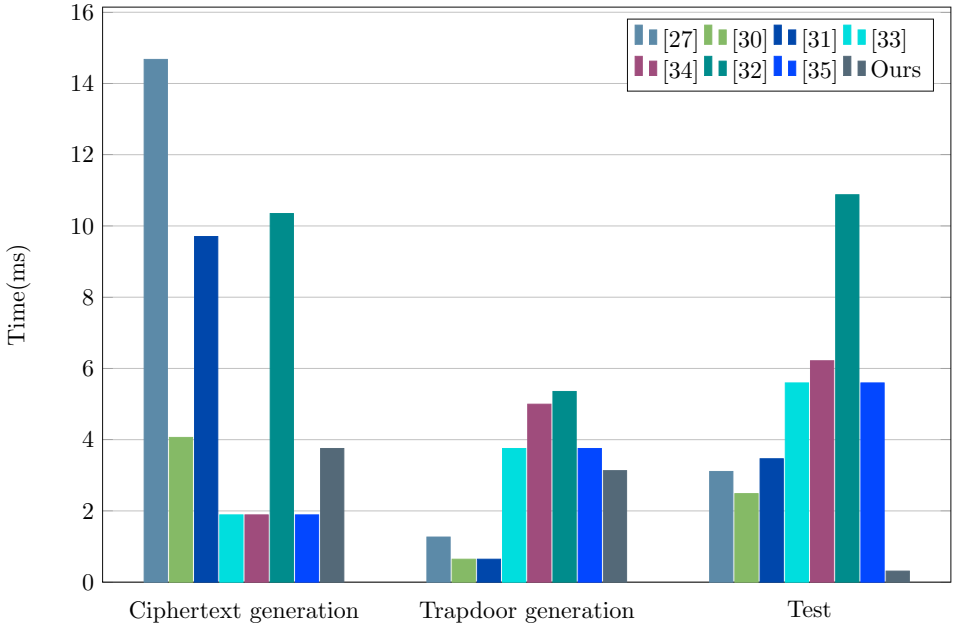

Fig. 2. Computation cost comparison.

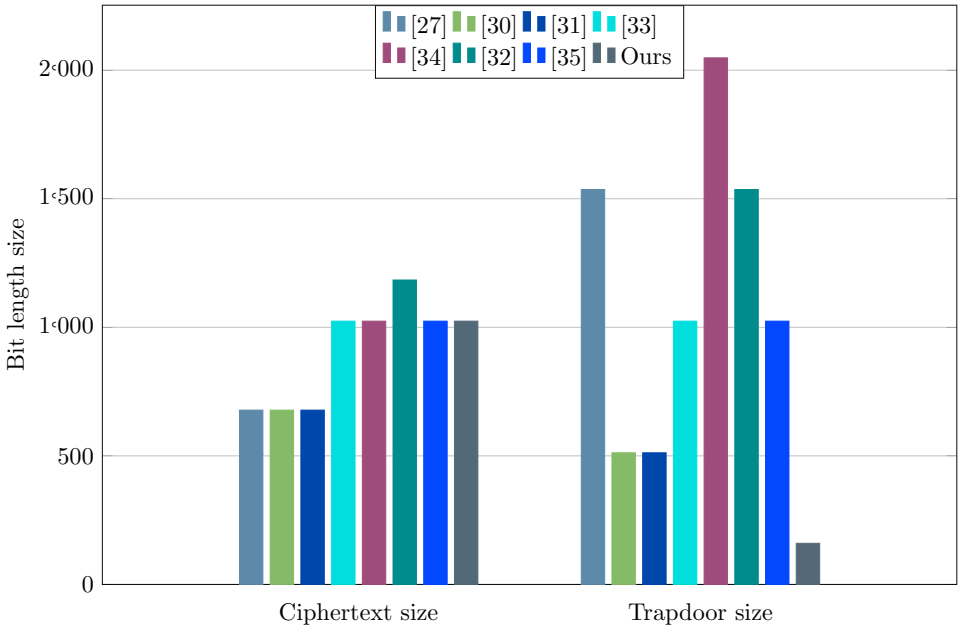

Fig. 3. Communication cost comparison.

7. Conclusion

The integration of IIoT and cloud computing has been very beneficial to the industrial sector in terms of reducing computing overhead. Security challenges however emerge due to the untrusted nature of the cloud servers. CLAEKS techniques are used to simultaneously provide the privacy, searchability, sharability and authentication of the ciphertexts uploaded on the cloud. However, it is shown in the published literature that none of the existing CLAEKS schemes provide the needed security requirements. To address this problem, in this paper, we enhance the security model of CLAEKS schemes and propose a new scheme with provable security in the enhanced security model. Comparisons with existing related schemes demonstrate that the proposed scheme is the only scheme that is secure in the enhanced security model. Computational and communication complexities show that security is achieved at some affordable cost.

Declaration of Competing Interest

There is no conflict of interest to report.

CRediT authorship contribution statement

Nasrollah Pakniat: Conceptualization, Methodology, Writing - review & editing. **Danial Shiraly:** Methodology, Writing - original draft. **Ziba Eslami:** Supervision, Writing - review & editing.

References

[1] Wallin L-O, Zimmerman T. 2017 Strategic roadmap for IoT network technology. Tech rep.. Gartner; 2017.
[2] Alaybeyi SB. Pragmatic strategies to improve industrial IoT security. tech rep.. Gartner; 2016.
[3] Song D, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Proceeding 2000 IEEE symposium on security and privacy. S&P 2000. IEEE; 2000. p. 44–55.
[4] Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public key encryption with keyword search. In: International conference on the theory and applications of cryptographic techniques. Springer; 2004. p. 506–22.
[5] Baek J, Safavi-Naini R, Susilo W. Public key encryption with keyword search revisited. In: International conference on computational science and its applications. Springer; 2008. p. 1249–59.
[6] Rhee H, Park J, Susilo W, Lee D. Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. 2010;83(5):763–71.
[7] Byun J, Rhee H, Park H, Lee D. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Workshop on secure data management. Springer; 2006. p. 75–83.
[8] Yau W, Heng S, Goi B. Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: International conference on autonomic and trusted computing. Springer; 2008. p. 100–5.
[9] Pakniat N. Public key encryption with keyword search and keyword guessing attack: a survey. In: 13th International Iranian society of cryptology conference on information security and cryptology (ISCISC), September 7–9, 2016 Tehran, Iran; 2016. p. 1–4.
[10] Rhee H, Park J, Lee D. Generic construction of designated tester public-key encryption with keyword search. Inf Sci 2012;205(Supplement C):93–109.
[11] Fang L, Susilo W, Ge C, Wang J. Public key encryption with keyword search secure against keyword guessing attacks without random oracle. Inf Sci 2013;238(Supplement C):221–41.
[12] Zhou Y, Guo H, Wang F, Luo W. Multi-key searchable encryption with designated server. Intell Autom Soft Comput 2016;22(2):295–301.
[13] Yang Y, Ma M. Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. IEEE Trans Inf Forensics Secur 2016;11(4):746–59.
[14] Lu Y, Wang G, Li J, Shen J. Efficient designated server identity-based encryption with conjunctive keyword search. Ann Telecommun 2017;72(5):359–70.
[15] Li Z, Zhao M, Jiang H, Xu Q. Multi-user searchable encryption with a designated server. Ann Telecommun 2017;72(9):617–29.
[16] Li C, Lee C, Shen J. An extended chaotic maps-based keyword search scheme over encrypted data resist outside and inside keyword guessing attacks in cloud storage services. Nonlin Dyn 2015;80(3):1601–11.
[17] Noroozi M, Eslami Z, Pakniat N. Comments on a chaos-based public key encryption with keyword search scheme. Nonlin Dyn 2018;94(2):1127–32.
[18] Huang Q, Li H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf Sci 2017;403:1–14.
[19] Noroozi M, Eslami Z. Public key authenticated encryption with keyword search: revisited. IET Inf Secur 2019;13(6):336–42.
[20] Qin B, Chen Y, Huang Q, Liu X, Zheng D. Public-key authenticated encryption with keyword search revisited: security model and constructions. Inf Sci 2020;516:515–28.
[21] Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, et al. Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. J Cryptol. 2008;21(3):350–91.
[22] Wu T, Tsai T, Tseng Y. Efficient searchable ID-based encryption with a designated server. Ann Telecommun 2014;69(7):391–402.
[23] Wang X, Mu Y, Chen R, Zhang X. Secure channel free ID-based searchable encryption for peer-to-peer group. J Comput Sci Technol 2016;31(5):1012–27.
[24] Noroozi M, Karoubi I, Eslami Z. Designing a secure designated server identity-based encryption with keyword search scheme: still unsolved. Ann Telecommun 2018;73(11):769–76.
[25] Li H, Huang Q, Shen J, Yang G, Susilo W. Designated-server identity-based authenticated encryption with keyword search for encrypted emails. Inf Sci 2019;481:330–43.
[26] Al-Riyami S, Paterson K. Certificateless public key cryptography. In: International conference on the theory and application of cryptology and information security. Springer; 2003. p. 452–73.
[27] Yanguo P, Jiangtao C, Changgen P, Zuobin Y. Certificateless public key encryption with keyword search. China Commun 2014;11(11):100–13.
[28] Wu T, Meng F, Chen C, Liu S, Pan J. On the security of a certificateless searchable public key encryption scheme. In: Pan J, Lin J, Wang C, Jiang X, editors. Genetic and evolutionary computing: proceedings of the tenth international conference on genetic and evolutionary computing, November 7–9, 2016 Fuzhou City, Fujian Province, China; 2017. p. 113–19.
[29] Zheng Q, Li X, Azgin A. CLKS: Certificateless keyword search on encrypted data. Springer; 2015. p. 239–53.
[30] Ma M, He D, Khan M, Chen J. Certificateless searchable public key encryption scheme for mobile healthcare system. Comput Electr Eng 2018;65:413–24.
[31] Ma M, He D, Kumar N, Choo K, Chen J. Certificateless searchable public key encryption scheme for industrial internet of things. IEEE Trans Ind Inf 2017;14(2):759–67.
[32] Pakniat N. Designated tester certificateless encryption with keyword search. J Inform Secur Appl 2019;49:102394.
[33] He D, Ma M, Zeadally S, Kumar N, Liang K. Certificateless public key authenticated encryption with keyword search for industrial internet of things. IEEE Trans Ind Inf 2018;14(8):3618–27.
[34] Wu L, Zhang Y, Ma M, Kumar N, He D. Certificateless searchable public key authenticated encryption with designated tester for cloud-assisted medical internet of things. Ann Telecommun 2019;74(7):423–34.
[35] Liu X, Li H, Yang G, Susilo W, Tonien J, Huang Q. Towards enhanced security for certificateless public-key authenticated encryption with keyword search. In: Steinfeld R, Yuen T, editors. Provable Security; 2019. p. 113–29. Cham
[36] Pakniat N., Eslami Z. Security flaws in certificateless public key authenticated encryption with keyword search schemes. Ann Telecommun submitted.
[37] Eslami Z, Pakniat N. Certificateless aggregate signcryption: security model and a concrete construction secure in the random oracle model. J King Saud Univ 2014;26(3):276–86.
[38] Libert B, Quisquater J-J. Identity based undeniable signatures. In: Cryptographers track at the RSA conference. Springer; 2004. p. 112–25.
[39] Yu Y, Xu C, Zhang X, Liao Y. Designated verifier proxy signature scheme without random oracles. Comput Math Appl 2009;57(8):1352–64.
[40] Lu Y, Li J. Efficient searchable public key encryption against keyword guessing attacks for cloud-based EMR systems. Cluster Comput 2019;22(1):285–99.
[41] Lynn B. PBC library: the pairing-based cryptography library. http://crypto.stanford.edu/pbc/ (2013) Accessed 1 April 2014,2014.