Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Constructing pairing-free certificateless public key encryption with keyword search*

Yang LU†‡, Ji-guo LI

College of Computer and Information, Hohai University, Nanjing 211100, China

†E-mail: luyangnsd@163.com

Received Aug. 11, 2017; Revision accepted Oct. 12, 2017; Crosschecked Aug. 15, 2019

Abstract: Searchable public key encryption enables a storage server to retrieve the publicly encrypted data without revealing the original data contents. It offers a perfect cryptographic solution to encrypted data retrieval in encrypted data storage systems. Certificateless cryptography (CLC) is a novel cryptographic primitive that has many merits. It overcomes the key escrow problem in identity-based cryptosystems and the cumbersome certificate problem in conventional public key cryptosystems. Motivated by the appealing features of CLC, three certificateless encryption with keyword search (CLEKS) schemes were presented in the literature. However, all of them were constructed with the costly bilinear pairing and thus are not suitable for the devices that have limited computing resources and battery power. So, it is interesting and worthwhile to design a CLEKS scheme without using bilinear pairing. In this study, we put forward a pairing-free CLEKS scheme that does not exploit bilinear pairing. We strictly prove that the scheme achieves keyword ciphertext indistinguishability against adaptive chosen-keyword attacks under the complexity assumption of the computational Diffie-Hellman problem in the random oracle model. Efficiency comparison and the simulation show that it enjoys better performance than the previous pairing-based CLEKS schemes. In addition, we briefly introduce three extensions of the proposed CLEKS scheme.

Key words: Searchable public key encryption; Certificateless public key encryption with keyword search; Bilinear pairing; Computational Diffie-Hellman problem
https://doi.org/10.1631/FITEE.1700534 **CLC number:** TP309

1 Introduction

With the speedy development and widespread application of cloud computing, increasingly more users are beginning to use the cloud for online data storing and sharing. To protect the confidentiality and privacy of personal data, a user may encrypt his/her data. Because the cloud storage server does not know the corresponding decryption keys, the confidentiality of user data is guaranteed. However, the conventional encryption paradigm makes it difficult to flexibly retrieve the encrypted data in cloud storage systems. One common solution is that a user downloads all his/her encrypted data from the cloud storage server and then decrypts them, regardless of what data he/she is searching for. Obviously, this is extremely inefficient due to the high cost of network transmission and heavy overhead at the user devices.

To solve this problem, searchable encryption (SE) as an efficient solution was introduced (Song et al., 2000; Boneh et al., 2004). Song et al. (2000) brought forth the concept of SE and presented a concrete searchable symmetric encryption (SSE) scheme. SE enables an untrustworthy server to retrieve the encrypted data while keeping the data contents secret.

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61772009 and U1736112), the Fundamental Research Funds for the Central Universities, China (Nos. 2016B10114 and 2017B17014), and the Natural Science Foundation of Jiangsu Province, China (No. BK20181304)

ORCID: Yang LU, http://orcid.org/0000-0003-4860-8384

Therefore, it offers a promising solution to encrypted data retrieval. Till now, many SSE schemes have been presented (Golle et al., 2004; Byun et al., 2006; Fu et al., 2015, 2016). However, the SSE schemes inevitably suffer from the key distribution problem due to the property of symmetric cryptography.

At Eurocrypt'04, Boneh et al. (2004) presented the notion of public key encryption with keyword search (PEKS). PEKS enables the users to make the keyword search queries on the data ciphertexts produced by standard public key encryption (PKE). In PEKS, three kinds of entities are involved: a sender, a server (also called a "tester"), and a receiver. To make the data ciphertexts searchable, a sender exploits a PKE scheme to encrypt the data to be sent and a PEKS scheme to produce a keyword ciphertext that is then appended to the encrypted data. To make the search queries on the encrypted data, a receiver generates and sends the server a keyword trapdoor in which the searched keyword is encoded. After receiving the keyword trapdoor, the server can test whether the keyword and keyword ciphertext correspond to the same keyword without seeing the plaintext of the encrypted data and the keyword. Finally, the server sends the receiver all encrypted data that contain the searched keyword. Inspired by Boneh et al. (2004)'s pioneering work, PEKS has aroused great attention in the research community. So far, numerous PEKS schemes and variants have been presented in the literature (Park et al., 2004; Hwang and Lee, 2007; Baek et al., 2008; Fang et al., 2009; Rhee et al., 2010; Zhang and Zhang, 2011; Hu and Liu, 2012; Dong et al., 2013; Lv et al., 2014; Guo and Yau, 2015; Shao and Yang, 2015). However, the abovementioned schemes were constructed under conventional public key cryptography (PKC) and thus inevitably suffer from the cumbersome certificate problem.

To solve this issue, Abdalla et al. (2008) introduced SE into identity-based cryptography (IBC) (Shamir, 1984) and presented the framework of identity-based encryption with keyword search (IBEKS). Compared with PEKS, IBEKS has the merit of no certificate. Abdalla et al. (2008) provided a general construction of IBEKS to show that IBEKS can be built from anonymous hierarchical identity-based encryption. Subsequently, several IBEKS schemes have been proposed (Tian and Wang, 2008; Siad, 2012; Wu et al., 2014; Tomida et al., 2015; Wang et al., 2016; Liu et al., 2017; Lu et al., 2017). In addition, searchable encryption was introduced into attribute-based encryption to build searchable attribute-based encryption (SABE) (Zheng et al., 2014; Sun et al., 2016; Li et al., 2017a, 2017b) to extend IBEKS by associating ciphertexts and trapdoors with the sets of descriptive attributes. However, in both IBEKS and SABE, all users' private keys are issued by a completely trusted private key generator (PKG), which causes the key escrow problem.

To address the problems imposed on the previous searchable schemes, Peng et al. (2014) proposed the framework of certificateless encryption with keyword search (CLEKS) that follows the idea of certificateless cryptography (CLC) introduced by Al-Riyami and Paterson (2003). CLC lies between conventional PKC and IBC, but preserves the certificateless merit. When using CLC, a user should first request a partial private key from a trusted key generation center (KGC). Then, the user randomly picks a secret and combines it with his/her partial private key to produce his/her private key. The public key is also calculated from the secret. Since the user-chosen secret is unknown to the KGC, CLC avoids the key escrow problem. To our knowledge, there exist three CLEKS schemes (Peng et al., 2014; Zheng et al., 2015; Islam et al., 2017) so far. Peng et al. (2014) proposed the first CLEKS scheme under the complexity assumption of the generalized bilinear Diffie-Hellman problem. Later, Zheng et al. (2015) constructed a CLEKS scheme under the complexity assumption of the decisional lineal problem. Recently, Islam et al. (2017) proposed a designated server CLEKS scheme that is secure under the bilinear Diffie-Hellman assumption and computational Diffie-Hellman assumption.

The goal of this study is to design a CLEKS scheme without using bilinear pairing. In practice, cryptographic operations are often performed on some devices with constrained resources, such as smart phone and personal digital assistant (PDA). Due to the limited computation performance or battery power, only the lightweight or power-saving cryptographic schemes can be employed on these devices. All the previous three CLEKS schemes (Peng et al., 2014; Zheng et al., 2015; Islam et al.,

2017) have to be based on the costly bilinear pairing. In practical implementations, bilinear pairing is less efficient than other common cryptographic operations and will greatly increase the overload of the systems. Therefore, a CLEKS scheme without using bilinear pairing would be more attractive in terms of computational cost.

In this study, we design a pairing-free CLEKS scheme. Under the complexity assumption of the computational Diffie-Hellman problem, we prove that the scheme achieves keyword ciphertext indistinguishability against adaptive chosen-keyword attacks using the random oracle model (Bellare and Rogaway, 1993; Canetti et al., 2004). Our scheme avoids the time-consuming pairing operations. Compared with the previous pairing-based CLEKS schemes (Peng et al., 2014; Zheng et al., 2015; Islam et al., 2017), it has obvious advantages in terms of both computational and communicational performances. Additionally, we show three extensions of the proposed scheme, including multi-receiver, conjunctive-keyword, and designated server CLEKS.

2 Preliminaries

2.1 Notations

In Table 1, we list the notations used throughout this paper and their descriptions.

2.2 Complexity assumption

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

Definition 1 (CDH problem) Assume that G is an elliptic curve group of prime order q. Given a generator P of G and a binary tuple $(xP, yP)\in G^2$ for unknown $x, y\in Z_q^*$, the CDH problem in G is to calculate $xyP\in G$.

The complexity assumption of the CDH problem states that the advantage $\mathrm{Adv}(A)$ is negligible for any polynomial-time algorithm A, where $\mathrm{Adv}(A)=\Pr[A(q, G, P, xP, yP)=xyP|P\in G\wedge x, y\in Z_q^*]$.

2.3 Framework and security model of CLEKS

In this study, a CLEKS scheme is composed of six algorithms:

Table 1 Notations and descriptions

Notation	Description
p, q	Two large prime numbers
F_p	A finite field of integers modulo p
$E(F_p)$	An elliptic curve over the finite field F_p
G	A group over the elliptic curve $E(F_p)$
P	A generator of group G
Z_q^*	The field of integers modulo q
H_1, H_2, H_3	Three cryptographic hash functions
A	An algorithm to solve the CDH problem
B	An adversary against CLEKS
psp	The set of public system parameters
mk	KGC's master key
ID_U	A user U's identity
PPK_U, PSK_U	A user U's partial public and private keys
PK_U, SK_U	A user U's public and private keys
$\mathrm{PK}_U^{\mathrm{f}}$, $\mathrm{SK}_U^{\mathrm{f}}$	The false public and private keys for user U chosen by adversary B
$\mathrm{ID}_{\mathrm{ch}}$	The challenge identity
$\mathrm{PK}_{\mathrm{ch}}$, $\mathrm{SK}_{\mathrm{ch}}$	The public and private keys according to challenge identity $\mathrm{ID}_{\mathrm{ch}}$
w	A keyword
C_w	The keyword ciphertext of keyword w
T_w	The keyword trapdoor of keyword w
θ	A random bit
θ'	The adversary B's guess for bit θ

1. Setup(λ)

Given a security parameter λ, it produces the KGC's master key mk and a set of public system parameters psp. This algorithm is executed by a KGC.

2. PartialKeyGen(psp, mk, ID_U)

Given the set of public system parameters psp, KGC's master key mk, and a user U's identity ID_U, it produces a partial public key PPK_U and a partial private key PSK_U. This algorithm is executed by the KGC.

3. UserKeyGen(psp, ID_U, PPK_U, PSK_U)

Given the set of public system parameters psp, a user U's identity ID_U, and his/her partial public and private keys (PPK_U, PSK_U), it produces a pair of full public and private keys (PK_U, SK_U).

4. Encrypt(psp, w, ID_U, PK_U)

Given the set of public system parameters psp, a keyword w, a user U's identity ID_U, and his/her public key PK_U, it produces a keyword ciphertext C_w.

5. Trapdoor(psp, w, ID_U, SK_U)

Given the set of public system parameters psp, a keyword w, a user U's identity ID_U, and his/her private key SK_U, it produces a keyword trapdoor T_w.

6. Test(psp, C_w, $T_{w'}$)

Given the set of public system parameters psp, a keyword ciphertext C_w, and a keyword trapdoor $T_{w'}$, it outputs 1 if C_w and $T_{w'}$ correspond to a same keyword (i.e., w=w') or 0 otherwise.

Definition 2 (Correctness) A CLEKS scheme Π=(Setup, PartialKeyGen, UserKeyGen, Encrypt, Trapdoor, Test) is said to be correct if for any keyword w, we have C_w=Encrypt(psp, w, ID_U, PK_U), T_w=Trapdoor(psp, w, ID_U, SK_U), and then Test(psp, C_w, T_w)=1, where psp and user U's public/private keys (PK_U, SK_U) are correctly produced according to the specification of the algorithms Setup, PartialKeyGen, and UserKeyGen.

As introduced in Peng et al. (2014) and Zheng et al. (2015), two distinct adversaries should be involved in the security model of CLEKS, namely, Type 1 adversary and Type 2 adversary. The Type 1 adversary models an inside storage server or an outside attacker who cannot access KGC's master key mk, but is allowed to replace public keys. The Type 2 adversary models an honest-but-curious KGC who possesses the master key mk, but is disallowed to replace public keys.

To formalize the security model of CLEKS, we define five oracles that are used to simulate potential attacking scenarios. A Type 1 or Type 2 adversary (denoted by B) is allowed to query some of these oracles adaptively. These oracles are simulated by a challenger (or a game simulator) as follows:

1. Oracle $O^{\mathrm{CreateUser}}$

For the identity ID_U input, a public key PK_U is responded if ID_U has already been created. Else, the identity ID_U is created by producing a private/public key pair (SK_U, PK_U) and then PK_U is returned. We assume that the following four oracles respond only to a user's identity ID_U that has been created.

2. Oracle $O^{\mathrm{PrivateKey}}$

For the identity ID_U input, a private key SK_U is responded.

3. Oracle $O^{\mathrm{PartialPrivateKey}}$

For the identity ID_U input, a partial private key PSK_U is responded. The Type 2 adversary does not need to query this oracle since it can calculate any user's partial private key using the KGC's master key mk.

4. Oracle $O^{\mathrm{ReplaceKey}}$

For the identity ID_U input and a false public key $\mathrm{PK}_U^{\mathrm{f}}$, the current public key associated with identity ID_U is replaced by $\mathrm{PK}_U^{\mathrm{f}}$. Here, adversary B is asked to supply the private key $\mathrm{SK}_U^{\mathrm{f}}$ that matches the false public key $\mathrm{PK}_U^{\mathrm{f}}$. This restriction is imposed as the challenger may not know the corresponding private key if a public key has been replaced. This oracle is queried merely by the Type 1 adversary.

5. Oracle O^{Trapdoor}

For inputting a keyword w and a user's identity ID_U, a keyword trapdoor T_w is responded. For the Type 1 adversary, if it has replaced the public key of the identity ID_U, the challenger should produce a keyword trapdoor T_w using the private key submitted by the adversary.

A CLEKS scheme must satisfy the keyword ciphertext indistinguishability against adaptive chosen-keyword attacks (hereafter referred to as "IND-CKA security" for short), which ensures that an adversary cannot distinguish the keyword ciphertexts of two different keywords on which it wants to be challenged. IND-CKA security can be defined by an adversarial game IND-CKA-Game played between a challenger and a Type 1/Type 2 adversary B as follows:

1. Setup

Given a security parameter λ, the challenger simulates algorithm Setup(λ) to produce a set of public system parameters psp and KGC's master key mk. It then provides adversary B with psp if it is a Type 1 adversary or both psp and mk if it is a Type 2 adversary.

2. Phase 1

Adversary B adaptively queries the oracles $\{O^{\mathrm{CreateUser}}, O^{\mathrm{PrivateKey}}, O^{\mathrm{PartialPrivateKey}}, O^{\mathrm{ReplaceKey}}, O^{\mathrm{Trapdoor}}\}$ if it is a Type 1 adversary or the oracles $\{O^{\mathrm{CreateUser}}, O^{\mathrm{PrivateKey}}, O^{\mathrm{Trapdoor}}\}$ if it is a Type 2 adversary.

3. Challenge

Adversary B outputs an identity $\mathrm{ID}_{\mathrm{ch}}$ and two different keywords (w_0, w_1). The challenger selects a random bit $\theta \in \{0, 1\}$ and calculates a challenge keyword ciphertext C_{w_θ}=Encrypt(psp, w_q, $\mathrm{ID}_{\mathrm{ch}}$, $\mathrm{PK}_{\mathrm{ch}}$). It then returns C_{w_θ} to adversary B.

4. Phase 2

Adversary B continues to make a sequence of oracle queries.

5. Guess

Eventually, adversary B gives its guess $\theta' \in \{0, 1\}$. Adversary B wins if $\theta=\theta'$ and the following conditions are simultaneously satisfied: (1) It has never queried oracle O^{Trapdoor} on either w_0 or w_1; (2) It has never queried either oracle $O^{\text{PrivateKey}}$ or oracle $O^{\text{PartialPrivateKey}}$ on ID_{ch} if it is a Type 1 adversary; (3) It has never queried the oracle $O^{\text{PrivateKey}}$ on ID_{ch} if it is a Type 2 adversary.

The advantage of adversary B in winning the above adversarial game is defined to be

$$\text{Adv}(B) = \left|2\Pr[\theta = \theta'] - 1\right|. \tag{1}$$

Definition 3 (IND-CKA security) We say that a CLEKS scheme achieves IND-CKA security if advantage $\text{Adv}(B)$ is negligible for any polynomial-time adversary B.

3 The proposed CLEKS scheme

3.1 Description of the scheme

The proposed pairing-free CLEKS scheme is described as follows:

1. Setup(λ)

This algorithm produces a cyclic group G with order q over an elliptic curve $E(F_p)$. Group G has the form $\{(x, y)|x, y\in F_p \wedge y^2=x^3+ax+b\}\cup\{O\}$, where a and b belong to the finite field F_p and satisfy $4a^3+27b^2\neq 0$ and O is a point at infinity. Let P be a generator of G. It then randomly chooses $s \in Z_q^*$ and computes $P_{\text{pub}}=sP$. Furthermore, it selects three hash functions H_1, H_2: $\{0, 1\}^* \to Z_q^*$ and H_3: $G \to Z_q^*$. Finally, it returns a set of public system parameters psp=$\{q, G, P, P_{\text{pub}}, H_1, H_2, H_3\}$ and KGC's master key mk=s.

2. PartialKeyGen(psp, mk, ID_U)

Given the set of system public parameters psp, KGC's master key mk=s, and a user U's identity ID_U, this algorithm randomly chooses $x \in Z_q^*$ and calculates a partial public key $\text{PPK}_U=xP$ and a partial private key $\text{PSK}_U=x+sH_1(\text{ID}_U)$ for user U.

3. UserKeyGen(psp, ID_U, PPK_U, PSK_U)

Given the set of system public parameters psp, a user U's identity ID_U, partial public key $\text{PPK}_U=xP$, and partial private key $\text{PSK}_U=x+sH_1(\text{ID}_U)$, this algorithm randomly chooses $y \in Z_q^*$, sets private key $\text{SK}_U=(\text{SK}_{U1}, \text{SK}_{U2})=(x+sH_1(\text{ID}_U), y)$, and calculates public key $\text{PK}_U=(\text{PK}_{U1}, \text{PK}_{U2})=(xP, yP)$.

4. Encrypt(psp, w, ID_U, PK_U)

Given the set of system public parameters psp, a keyword w, a user U's identity ID_U, and his/her public key $\text{PK}_U=(\text{PK}_{U1}, \text{PK}_{U2})$, this algorithm randomly selects $r \in Z_q^*$ and computes a keyword ciphertext $C_w=(C_{w1}, C_{w2})=(rP, H_3(rH_2(w)Q_U))$, where $Q_U=\text{PK}_{U1}+\text{PK}_{U2}+H_1(\text{ID}_U)P_{\text{pub}}$.

5. Trapdoor(psp, w, ID_U, SK_U)

Given the set of system public parameters psp, a keyword w, a user U's identity ID_U, and his/her private key $\text{SK}_U=(\text{SK}_{U1}, \text{SK}_{U2})$, this algorithm computes a keyword trapdoor $T_w=(\text{SK}_{U1}+\text{SK}_{U2})H_2(w)$.

6. Test(psp, C_w, $T_{w'}$)

Given the set of system public parameters psp, a keyword ciphertext $C_w=(C_{w1}, C_{w2})$, and a trapdoor $T_{w'}$, this algorithm checks whether $C_{w2}=H_3(T_{w'}C_{w1})$ holds. If it does, output 1; otherwise, output 0.

3.2 Correctness proof

Theorem 1 (Correctness) The proposed pairing-free CLEKS scheme is correct.

Proof According to the above scheme description, we have

$$\begin{aligned} H_3(T_{w'}C_{w1}) &= H_3((\text{SK}_{U1}+\text{SK}_{U2})H_2(w')rP) \\ &= H_3((x+sH_1(\text{ID}_U)+y)H_2(w')rP) \\ &= H_3(rH_2(w')(\text{PK}_{U1}+\text{PK}_{U2}+H_1(\text{ID}_U)P_{\text{pub}})) \\ &= H_3(rH_2(w')Q_U)). \end{aligned} \tag{2}$$

Clearly, if $w=w'$, then $C_{w2}=H_3(T_{w'}C_{w1})$. Therefore, the scheme is correct.

3.3 Security proof

Theorem 2 (IND-CKA security) The proposed pairing-free CLEKS scheme achieves IND-CKA security under the complexity assumption of the CDH problem in the random oracle model.

This theorem can be proved by the following Lemmas 1 and 2.

Lemma 1 Let H_1, H_2, and H_3 be three random oracles. If there is a Type 1 adversary B who can win the game IND-CKA-Game with advantage ε and running time τ, then there exists an algorithm A that can solve the CDH problem with advantage

$$\varepsilon' \geq \frac{\varepsilon}{q_3(q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}+1)e} \tag{3}$$

and running time $\tau' \leq \tau+(q_1+q_2+q_3+q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{RK}})O(1)+q_{\mathrm{C}}(3\tau_1+O(1))+q_{\mathrm{T}}(\tau_2+O(1))$, where q_{C}, q_{SK}, q_{PSK}, q_{RK}, q_{T}, q_1, q_2, and q_3 denote the maximum number of adversary B's queries to oracles $O^{\mathrm{CreateUser}}$, $O^{\mathrm{PrivateKey}}$, $O^{\mathrm{PartialPrivateKey}}$, $O^{\mathrm{ReplaceKey}}$, O^{Trapdoor}, H_1, H_2, and H_3, respectively, e denotes the Euler number, and τ_1 and τ_2 denote the time for computing a scalar multiplication in G and a modular multiplication in Z_q^*, respectively.

Proof Let (q, G, P, aP, bP) be a CDH problem instance given to algorithm A. To compute abP, algorithm A simulates a challenger and plays the game IND-CKA-Game with adversary B as follows:

1. Setup

Algorithm A first sets $P_{\mathrm{pub}}=aP$ and then provides adversary B with the set of public system parameters $\mathrm{psp}=\{q, G, P, P_{\mathrm{pub}}, H_1, H_2, H_3\}$, where H_1, H_2, and H_3 are three random oracles.

2. Phases 1 and 2

During two query-answer phases, adversary B can query oracles $O^{\mathrm{CreateUser}}$, $O^{\mathrm{PrivateKey}}$, $O^{\mathrm{PartialPrivateKey}}$, $O^{\mathrm{ReplaceKey}}$, O^{Trapdoor}, H_1, H_2, and H_3 in an adaptive manner. Algorithm A answers adversary B's various queries in the following way:

(1) H_1 queries

To answer these queries, algorithm A keeps an H_1-table that is composed of tuples (ID_i, h_{1i}). On receiving an identity ID_i, algorithm A is executed as follows:

1) If identity ID_i has been contained in a tuple (ID_i, h_{1i}) on the H_1-table, it outputs h_{1i} directly.

2) Otherwise, it randomly selects $h_{1i} \in Z_q^*$, adds a new tuple (ID_i, h_{1i}) to the H_1-table, and then outputs h_{1i}.

(2) H_2 queries

To answer these queries, algorithm A keeps an H_2-Table that is composed of the tuples (w_i, h_{2i}). On receiving a keyword w_i, algorithm A executes as follows:

1) If the keyword w_i has been contained in a tuple (w_i, h_{2i}) on the H_2-Table, it outputs h_{2i} directly.

2) Otherwise, it randomly selects $h_{2i} \in Z_q^*$, adds a new tuple (w_i, h_{2i}) to the H_2-Table, and then outputs h_{2i}.

(3) H_3 queries

To answer these queries, algorithm A keeps a H_3-table that is composed of tuples (K_i, h_{3i}). On receiving a value $K_i \in G$, algorithm A is executed as follows:

1) If value K_i has been contained in a tuple (K_i, h_{3i}) on the H_3-table, it outputs h_{3i} directly.

2) Otherwise, it randomly selects $h_{3i} \in Z_q^*$, adds a new tuple (K_i, h_{3i}) to the H_3-table, and then outputs h_{3i}.

(4) $O^{\mathrm{CreateUser}}$ queries

To answer these queries, algorithm A keeps a User-table that is composed of tuples $(\mathrm{ID}_i, \mathrm{PK}_i, \mathrm{SK}_i, x_i, y_i, c_i)$. On receiving an identity ID_i, algorithm A checks whether identity ID_i has been contained in a tuple $(\mathrm{ID}_i, \mathrm{PK}_i, \mathrm{SK}_i, x_i, y_i, c_i)$ on the User-table. If so, it outputs PK_i directly. Otherwise, algorithm A picks a random $c_i \in \{0, 1\}$ so that $\Pr[c_i=1]=\gamma$ where γ will be determined later and is performed as follows:

1) If $c_i=1$, it first randomly chooses $x_i, y_i \in Z_q^*$ and computes $\mathrm{PK}_i=(x_iP, y_iP)$. It then adds a new tuple $(\mathrm{ID}_i, \mathrm{PK}_i, \perp, x_i, y_i, 1)$ to the User-table and outputs PK_i.

2) If $c_i=0$, it randomly chooses $y_i, s_i, z_i \in Z_q^*$, sets $\mathrm{SK}_i=(z_i, y_i)$, and computes $\mathrm{PK}_i=(z_iP-s_iP_{\mathrm{pub}}, y_iP)$. It then adds new tuples $(\mathrm{ID}_i, \mathrm{PK}_i, \mathrm{SK}_i, \perp, \perp, 0)$ and (ID_i, s_i) into the User-table and H_1-table, respectively, and returns PK_i.

(5) $O^{\mathrm{PrivateKey}}$ queries

When receiving an identity ID_i, algorithm A searches for identity ID_i in the User-table to find a tuple $(\mathrm{ID}_i, \mathrm{PK}_i, \mathrm{SK}_i, x_i, y_i, c_i)$. It aborts if $c_i=1$ (this event is denoted by Event_1) or returns SK_i otherwise.

(6) $O^{\mathrm{PartialPrivateKey}}$ queries

When receiving an identity ID_i, algorithm A searches for identity ID_i in the User-table to find a tuple $(\mathrm{ID}_i, \mathrm{PK}_i, \mathrm{SK}_i, x_i, y_i, c_i)$. It aborts if $c_i=1$ (this event is denoted by Event_2) or returns SK_{i1} (the first component of private key SK_i) otherwise.

(7) $O^{\mathrm{ReplaceKey}}$ queries

When receiving an identity ID_i and a false public key K_i^{f}, algorithm A records the replacement.

(8) O^{Trapdoor} queries

When receiving an identity ID_i and a keyword w, algorithm A searches for identity ID_i in the User-table to find a tuple (ID_i, PK_i, SK_i, x_i, y_i, c_i).

1) If c_i=0 or PK_i has been replaced, algorithm A produces a trapdoor T_w=(SK_{i1}+SK_{i2})$H_2(w)$ since it knows the private key SK_i=(SK_{i1}, SK_{i2}) associated with identity ID_i.

2) Otherwise, it aborts (this event is denoted by Event_3).

3. Challenge

Adversary B outputs an identity $\mathrm{ID}_{\mathrm{ch}}$ and two different keywords w_0 and w_1. Algorithm A first retrieves a tuple of the form ($\mathrm{ID}_{\mathrm{ch}}$, $\mathrm{PK}_{\mathrm{ch}}$, $\mathrm{SK}_{\mathrm{ch}}$, x_{ch}, y_{ch}, c_{ch}) from the User-table.

(1) If $c_{\mathrm{ch}}\neq 1$, it aborts (this event is denoted by Event_4).

(2) Otherwise, it randomly chooses $\theta\in\{0, 1\}$ and $C_{w_\theta 2}\in Z_q^*$, sets $C_{w_\theta 1}=bP$, and then returns $C_{w_\theta}=(C_{w_\theta 1}, C_{w_\theta 2})$ as the challenge ciphertext to adversary B. Note that $C_{w_\theta 2}$ is implicitly defined as $H_3(bH_2(w_\theta)\cdot(\mathrm{PK}_{\mathrm{ch1}}+\mathrm{PK}_{\mathrm{ch2}}+H_1(\mathrm{ID}_{\mathrm{ch}})P_{\mathrm{pub}}))$, where $\mathrm{PK}_{\mathrm{ch1}}=x_{\mathrm{ch}}P$ and $\mathrm{PK}_{\mathrm{ch2}}=y_{\mathrm{ch}}P$.

4. Guess

Eventually, adversary B outputs its guess θ'. To solve the given CDH problem, algorithm A ignores adversary B's guess and retrieves a tuple of the form ($\mathrm{ID}_{\mathrm{ch}}$, $\mathrm{PK}_{\mathrm{ch}}$, $\mathrm{SK}_{\mathrm{ch}}$, x_{ch}, y_{ch}, c_{ch}) from the User-table, randomly selects a tuple (K, h_3) from the H_3-table, and then calculates $Z=H_1(\mathrm{ID}_{\mathrm{ch}})^{-1}H_2(w_\theta)^{-1}(K-x_{\mathrm{ch}}H_2(w_\theta)bP-y_{\mathrm{ch}}H_2(w_\theta)bP)$ as the answer to the CDH problem. If $K=bH_2(w_\theta)(\mathrm{PK}_{\mathrm{ch1}}+\mathrm{PK}_{\mathrm{ch2}}+H_1(\mathrm{ID}_{\mathrm{ch}})P_{\mathrm{pub}})$, then we can easily deduce that $Z=abP$.

Next, we derive the bound on the advantage of algorithm A. To do so, we define the following events:

(1) Ask$K$

Adversary B queries the random oracle H_3 on the value $K=bH_2(w_\theta)(\mathrm{PK}_{\mathrm{ch1}}+\mathrm{PK}_{\mathrm{ch2}}+H_1(\mathrm{ID}_{\mathrm{ch}})P_{\mathrm{pub}})$.

(2) Abort

Algorithm A aborts the game abnormally during the simulation.

According to the above simulation, it is clear that $\Pr[\neg\mathrm{Abort}]=\Pr[\neg\mathrm{Event}_1\wedge\neg\mathrm{Event}_2\wedge\neg\mathrm{Event}_3\wedge\neg\mathrm{Event}_4]$. Since we can obtain $\Pr[\neg\mathrm{Event}_1\wedge\neg\mathrm{Event}_2\wedge\neg\mathrm{Event}_3]=\gamma(1-\gamma)^{q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}}$ and $\Pr[\neg\mathrm{Event}_4]=\gamma$, we have $\Pr[\neg\mathrm{Abort}]=\gamma(1-\gamma)^{q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}}$, where q_{SK}, q_{PSK}, and q_{T} denote the maximum number of adversary B's queries to oracles $O^{\mathrm{PrivateKey}}$, $O^{\mathrm{PartialPrivateKey}}$, and O^{Trapdoor}, respectively. The value of $\gamma(1-\gamma)^{q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}}$ is maximized at $\gamma_{\max}=1/(q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}+1)$. So, we have

$$\Pr[\neg\mathrm{Abort}]\geq\frac{1}{(q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}+1)e}. \tag{4}$$

Let E^*=Ask$K$|¬Abort. Obviously, if E^* does not occur, then adversary B will not obtain an advantage greater than 1/2 in guessing bit θ. Therefore, $\Pr[\theta=\theta'|\neg E^*]=1/2$.

By splitting $\Pr[\theta=\theta']$, we have $\Pr[\theta=\theta']=\Pr[\theta=\theta'|\neg E^*]\Pr[\neg E^*]+\Pr[\theta=\theta'|E^*]\Pr[E^*]\leq\Pr[\neg E^*]/2+\Pr[E^*]=1/2+\Pr[E^*]/2$. Therefore, $|2\Pr[\theta=\theta']-1|\leq\Pr[E^*]$.

According to the advantage definition in the game IND-CKA-Game, we have $\varepsilon\leq\Pr[E^*]\leq\Pr[\mathrm{Ask}K]/\Pr[\neg\mathrm{Abort}]$. Thus, we have

$$\Pr[\mathrm{Ask}K]\geq\Pr[\neg\mathrm{Abort}]\varepsilon\geq\frac{\varepsilon}{(q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}+1)e}. \tag{5}$$

Once event Ask$K$ occurs, algorithm A can compute abP by selecting the correct tuple from the H_3-table probability $1/q_3$. Therefore, we obtain the bound on algorithm A's advantage:

$$\varepsilon'\geq\frac{1}{q_3}\Pr[\mathrm{Ask}K]\geq\frac{\varepsilon}{(q_{\mathrm{SK}}+q_{\mathrm{PSK}}+q_{\mathrm{T}}+1)e}. \tag{6}$$

Lemma 2 Let H_1, H_2, and H_3 be three random oracles. If there is a Type 2 adversary B who can win the game IND-CKA-Game with advantage ε and running time τ, then there must be an algorithm A that can solve the CDH problem with advantage

$$\varepsilon'\geq\frac{\varepsilon}{q_3(q_{\mathrm{SK}}+q_{\mathrm{T}}+1)e} \tag{7}$$

and running time $\tau'\leq\tau+(q_1+q_2+q_3+q_{\mathrm{SK}})O(1)+q_{\mathrm{C}}(2\tau_1+\tau_2+O(1))+q_{\mathrm{T}}(\tau_2+O(1))$, where q_{C}, q_{SK}, q_{T}, q_1, q_2, and q_3

denote the maximum numbers of adversary B's queries to oracles $O^{\text{CreateUser}}$, $O^{\text{PrivateKey}}$, O^{Trapdoor}, H_1, H_2, and H_3, respectively, e denotes the Euler number, and τ_1 and τ_2 denote the time for computing a scalar multiplication in G and a modular multiplication in Z_q^*, respectively.

Proof Let (q, G, P, aP, bP) be a CDH problem instance given to algorithm A. To compute abP, algorithm A simulates a challenger and plays the IND-CKA-Game with adversary B as follows:

1. Setup

Algorithm A first randomly selects $\alpha \in Z_q^*$ and calculates $P_{\text{pub}}=\alpha P$. It then provides adversary B with mk=α and psp=$\{q, G, P, P_{\text{pub}}, H_1, H_2, H_3\}$, where H_1, H_2, and H_3 are three random oracles.

2. Phases 1 and 2

During two query-answer phases, adversary B can query oracles $O^{\text{CreateUser}}$, $O^{\text{PrivateKey}}$, O^{Trapdoor}, H_1, H_2, and H_3 in an adaptive manner. Algorithm A answers adversary B's queries to oracles H_1–H_3 and $O^{\text{PrivateKey}}$ as in the proof of Lemma 1 and handles other queries as follows:

(1) $O^{\text{CreateUser}}$ queries

To answer these queries, algorithm A keeps a User-table that is composed of tuples (ID_i, PK_i, SK_i, t_i, y_i, c_i). On receiving an identity ID_i, algorithm A checks whether identity ID_i has been contained in a tuple on the User-table. If so, it outputs PK_i directly. Otherwise, it picks a random $c_i \in \{0, 1\}$ so that $\Pr[c_i=1]=\gamma$, where γ will be determined later and is performed as follows:

1) If $c_i=1$, it first randomly chooses $t_i, y_i \in Z_q^*$ and computes $\text{PK}_i=(\text{PK}_{i1}, \text{PK}_{i2})=(t_iaP, y_iP)$. It then adds a new tuple (ID_i, PK_i, $\perp$, t_i, y_i, 1) into the User-table and outputs PK_i.

2) If $c_i=0$, it randomly chooses $s_i, x_i, y_i \in Z_q^*$, sets $\text{SK}_i=(x_i+\alpha s_i, y_i)$, and computes $\text{PK}_i=(x_iP, y_iP)$. It then inserts new tuples (ID_i, PK_i, SK_i, $\perp$, $\perp$, 0) and (ID_i, s_i) into the User-table and H_1-table, respectively, and outputs PK_i.

(2) O^{Trapdoor}

On receiving an identity ID_i and a keyword w, algorithm A searches for identity ID_i in the User-table to find a tuple (ID_i, PK_i, SK_i, t_i, y_i, c_i).

1) If $c_i=1$, algorithm A aborts since it cannot answer the query.

2) Otherwise, it produces a trapdoor $T_w=(\text{SK}_{i1}+\text{SK}_{i2})H_2(w)$ since it knows the private key $\text{SK}_i=(\text{SK}_{i1}, \text{SK}_{i2})$ according to identity ID_i.

3. Challenge

Adversary B submits an identity ID_{ch} and two different keywords w_0 and w_1. Algorithm A first retrieves a tuple of the form (ID_{ch}, PK_{ch}, SK_{ch}, t_{ch}, y_{ch}, c_{ch}) from the User-table.

(1) If $c_{\text{ch}}\neq 1$, it aborts.

(2) Else, it chooses $\theta \in \{0, 1\}$ and $C_{w_\theta 2} \in Z_q^*$ randomly, sets $C_{w_\theta 1}=bP$, and then returns $C_{w_\theta}=(C_{w_\theta 1}, C_{w_\theta 2})$ as the challenge keyword ciphertext to adversary B. Note that $C_{w_\theta 2}$ is implicitly defined as $H_3(bH_2(w_\theta)(\text{PK}_{\text{ch1}}+\text{PK}_{\text{ch2}}+H_1(\text{ID}_{\text{ch}})P_{\text{pub}}))$, where $\text{PK}_{\text{ch1}}=t_{\text{ch}}aP$ and $\text{PK}_{\text{ch2}}=y_{\text{ch}}P$.

4. Guess

Eventually, adversary B outputs its guess θ'. To solve the given CDH problem, algorithm A ignores adversary B's guess and retrieves a tuple of the form (ID_{ch}, PK_{ch}, SK_{ch}, t_{ch}, y_{ch}, c_{ch}) from the User-table, randomly selects a tuple (K, h_3) from the H_3-table and then calculates $Z=(t_{\text{ch}})^{-1}H_2(w_\theta)^{-1}\cdot(K-y_{\text{ch}}H_2(w_\theta)bP-\alpha H_1(\text{ID}_{\text{ch}})H_2(w_\theta)bP)$ as the answer to the CDH problem. If $K=bH_2(w_\theta)(\text{PK}_{\text{ch1}}+\text{PK}_{\text{ch2}}+H_1(\text{ID}_{\text{ch}})P_{\text{pub}})$, then we can easily deduce that $Z=abP$.

Similarly, we can deduce that the bound on algorithm A_{CDH}'s advantage is $\varepsilon' \geq \dfrac{\varepsilon}{q_3(q_{\text{SK}}+q_{\text{T}}+1)e}$.

3.4 Efficiency evaluation and comparison

To evaluate the efficiency of the proposed pairing-free CLEKS scheme, we compare it with the previous pairing-based CLEKS schemes (Peng et al., 2014; Zheng et al., 2015; Islam et al., 2017). For ease of representation, we denote these CLEKS schemes by PCPY-CLEKS, ZLA-CLEKS, and IORA-CLEKS, separately. In addition, we consider only the single keyword encryption version of all compared schemes.

Five cryptographic operations are considered in the comparison (Table 2), where bilinear pairing is a map e: $G_1\times G_1\rightarrow G_2$. As usual, the costs of general cryptographic hash, point addition in either G_1 or G, and modular addition in Z_q^* are ignored.

The cost of an algorithm is measured by the sum of the costs of all cryptographic operations. For

Table 2 Notations in the efficiency comparison

Notation	Description
T_{BP}	Time cost for computing a bilinear pairing
T_{SM1}	Time cost for computing a scalar multiplication in bilinear group G_1
T_{SM}	Time cost for computing a scalar multiplication in elliptic curve group G
T_{MM}	Time cost for computing a modular multiplication in Z_q^*
T_{MH}	Time cost for computing a map-to-point hash
$\|G_1\|$	Bit length of an element in bilinear group G_1
$\|G\|$	Bit length of an element in elliptic curve group G
f	Bit-length of a hash value

example, to encrypt a keyword w, the keyword encryption algorithm Encrypt in our CLEKS scheme requires computing three scalar multiplications in G and one modular multiplication in Z_q^*. Therefore, the cost of the algorithm Encrypt is $3T_{SM}+T_{MM}$. In addition, the size of a keyword ciphertext/trapdoor is calculated by counting the total number of the group elements and hash values included. For example, a keyword ciphertext in our scheme is composed of two elements in G. Thus, its size is $2|G|$ bits. In the scheme PCPY-CLEKS, the symbol "f" denotes the bit-length of a hash value, which will be 256 bits if the SHA-256 hash function is used.

The details of the four schemes compared are given in Table 3.

We simulate these CLEKS schemes using the MIRACL library (MIRACL Ltd., 2012). The experimental platform is a laptop running Windows XP with 3-GHz Intel PIV CPU and 512-MB memory. For our pairing-free CLEKS scheme, to obtain the security level of 1024-bit RSA, we implement the elliptic curve group G using the security parameter secp160r1 recommended by Standards for Efficient Cryptography Group (2000). For the pairing-based CLEKS schemes, to obtain the same security level,Type A pairing over the supersingular elliptic curve $E(F_p)$: $y^2=x^3+x$ with embedding degree 2 is employed, where group size p is a 512-bit prime number satisfying $p+1=hr$ and h is a 160-bit Solinas prime number. The simulation results are given in Table 4.

According to Table 4, we can see that the keyword encryption algorithm of our scheme costs about 6.85 ms, which is about 4.0% of that of PCPY-CLEKS, 19.6% of that of ZLA-CLEKS, and 19.6% of that of IORA-CLEKS. The trapdoor algorithm of our CLEKS scheme costs about 0.22 ms, which is about 0.7% of that of PCPY-CLEKS, 0.4% of that of ZLA-CLEKS, and 2.3% of that of IORA-CLEKS. The keyword ciphertext size of our scheme is 1024 bits, which is slightly larger than that of PCPY-CLEKS, but is only 1/2 of that of ZLA-CLEKS and 2/3 of that of IORA-CLEKS.

In addition, the trapdoor size of our scheme is 160 bits, which is about 10.4% of that of PCPY-CLEKS, 7.8% of that of ZLA-CLEKS, and 31.2% of that of IORA-CLEKS. The simulation results show that our CLEKS scheme enjoys the best efficiency.

Table 3 Computation and communication costs of the compared CLEKS schemes

Scheme	Computation cost (ms)			Communication cost (bit)	
	Encrypt	Trapdoor	Test	Ciphertext size	Trapdoor size
PCPY-CLEKS	$7T_{BP}+4T_{SM1}+3T_{MH}$	$4T_{SM1}+T_{MH}$	$T_{BP}+2T_{SM}$	$\|G_1\|+f$	$3\|G_1\|$
ZLA-CLEKS	$5T_{SM1}+T_{MH}$	$8T_{SM1}+T_{MH}$	$4T_{BP}$	$4\|G_1\|$	$4\|G_1\|$
IORA-CLEKS	$5T_{SM1}+T_{MH}$	$T_{SM1}+T_{MH}$	$2T_{BP}+3T_{SM}$	$3\|G_1\|$	$\|G_1\|$
Ours	$3T_{SM}+T_{MM}$	T_{MM}	T_{SM}	$2\|G\|$	$\|Z_q^*\|$

Table 4 Simulation results of the compared CLEKS schemes

Scheme	Computation cost (ms)			Communication cost (bit)	
	Encrypt	Trapdoor	Test	Ciphertext size	Trapdoor size
PCPY-CLEKS	172.92	28.60	32.79	768	1536
ZLA-CLEKS	34.94	51.03	80.15	2048	2048
IORA-CLEKS	34.94	9.42	59.22	1536	512
Ours	6.85	0.22	2.21	1024	160

4 Three extensions of the proposed scheme

In this section, we present three extensions of the proposed CLEKS scheme. We skip the detailed descriptions of the schemes and show only the extended algorithms.

4.1 Multi-receiver CLEKS scheme

All existing CLEKS schemes consider only the single-receiver scenario. Using a single-receiver CLEKS scheme, a sender needs to repeatedly encrypt the same keyword associated with a data file with each receiver's public key if he/she wants to share the data file to a group of receivers. Obviously, it is inconvenient and less efficient.

A multi-receiver CLEKS scheme can effectively eliminate the repeated cryptographic operations and thus minimize the cost. To encrypt a keyword w for n receivers, the sender can perform the keyword encryption algorithm shown in Algorithm 1.

Algorithm 1 Keyword encryption algorithm in the multi-receiver CLEKS scheme

Encrypt(psp, w, $\{\mathrm{ID}_i, \mathrm{PK}_i=(\mathrm{PK}_{i1}, \mathrm{PK}_{i2})\}_{1\le i\le n}$)

Given the set of public system parameter psp, a keyword w, and multiple receivers' identities and public keys $\{\mathrm{ID}_i, \mathrm{PK}_i=(\mathrm{PK}_{i1}, \mathrm{PK}_{i2})_{1\le i\le n}\}$, this algorithm randomly selects $r\in Z_q^*$, computes $C_{w1}=rP$, and then uses each receiver i's identity and public key $\{\mathrm{ID}_i, \mathrm{PK}_i=(\mathrm{PK}_{i1}, \mathrm{PK}_{i2})\}_{1\le i\le n}$ to compute $C_{w2i}=H_3(rH_2(w)(\mathrm{PK}_{i1}+\mathrm{PK}_{i2}+H_1(\mathrm{ID}_i)P_{\mathrm{pub}})$. Finally, the multi-receiver keyword ciphertext will be $(C_{w1}, \{C_{w2i}\}_{1\le i\le n})$ and the keyword ciphertext used for the algorithm Test for receiver i will be (C_{w1}, C_{w2i}).

Other algorithms are the same as the ones in the original CLEKS scheme.

4.2 Conjunctive-keyword CLEKS scheme

All existing CLEKS schemes support only single-keyword search. However, single-keyword search often leads to far too coarse results. Therefore, the conjunctive-keyword search function becomes essential. Though Peng et al. (2014)'s and Islam et al. (2017)'s CLEKS schemes support multi-keyword encryption, they can deal with only one keyword in the trapdoor algorithm. To support conjunctive-keyword search, we extend our CLEKS scheme by encoding multiple keywords in both the keyword ciphertext and the trapdoor.

To encrypt a keyword set $W=\{w_i|i=1, 2, \ldots, n\}$ for a receiver U, the sender can perform the keyword encryption algorithm shown in Algorithm 2. On the receiver side, the receiver can perform the trapdoor algorithm shown in Algorithm 3 to generate a conjunctive-keyword trapdoor.

Algorithm 2 Keyword encryption algorithm in the conjunctive-keyword CLEKS scheme

Encrypt(psp, W, ID_U, PK_U)

Given the set of the public system parameter psp, a keyword set $W=\{w_i|i=1, 2, \ldots, n\}$, a user's identity ID_U, and public key $\mathrm{PK}_U=(\mathrm{PK}_{U1}, \mathrm{PK}_{U2})$, this algorithm randomly selects $r\in Z_q^*$, and computes a ciphertext $C_W=(C_{W1}, C_{W2})=(rP, H_3(r\sum_{i=1}^{n}H_2(w_i)Q_U))$, where $Q_U=\mathrm{PK}_{U1}+\mathrm{PK}_{U2}+H_1(\mathrm{ID}_U)P_{\mathrm{pub}}$.

Algorithm 3 Trapdoor algorithm in the conjunctive-keyword CLEKS scheme

Trapdoor(psp, W, ID_U, SK_U)

Given the set of the public system parameter psp, a keyword set $W=\{w_i|i=1, 2, \ldots, n\}$, a user's identity ID_U, and private key $\mathrm{SK}_U=(\mathrm{SK}_{U1}, \mathrm{SK}_{U2})$, this algorithm computes a trapdoor $T_W=(\mathrm{SK}_{U1}+\mathrm{SK}_{U2})\sum_{i=1}^{n}H_2(w_i)$.

When receiving a search query, the server can test whether or not a keyword ciphertext matches a keyword trapdoor as in the original CLEKS scheme.

4.3 Designated server CLEKS scheme

In our CLEKS scheme, a secure channel should be established between the server and each receiver to convey the keyword trapdoors. If not, an outside adversary who obtains trapdoor T_w can generate ciphertext $C_{w'}$ of a guessed keyword w' and then check whether Test(psp, $C_{w'}$, T_w)=1. Once keyword w encoded in trapdoor T_w is revealed, the adversary can further test whether a keyword ciphertext transmitted to the server is related to T_w. To remove the requirement of the secure channel, we extend our CLEKS scheme to a designated server CLEKS scheme, where a keyword ciphertext is generated under both a designated server's and a receiver's public keys, and only the designated server can execute the algorithm Test to verify whether a keyword ciphertext matches a keyword trapdoor using its private key.

In the designated server CLEKS scheme, a designated server S with identity ID_S first requests a partial private key $\mathrm{PSK}_S=x_S+sH_1(\mathrm{ID}_S)$ and a partial

public key $PPK_S=x_SP$ from the KGC and then produces its private key $SK_S=(SK_{S1}, SK_{S2})=(x_S+sH_1(ID_S), y_S)$ and public key $PK_S=(PK_{S1}, PK_{S2})=(x_SP, y_SP)$, where $x_S, y_S \in Z_q^*$.

To encrypt a keyword w for a receiver U, a sender can perform the keyword encryption algorithm shown in Algorithm 4. The receiver generates the trapdoor as in the original CLEKS scheme. When receiving a search query, the designated server uses its private key SK_S to identify whether a keyword ciphertext C_w matches a trapdoor $T_{w'}$ by executing the test algorithm shown in Algorithm 5.

Algorithm 4 Keyword encryption algorithm in the designated server CLEKS scheme

Encrypt(psp, w, ID_U, PK_U, ID_S, PK_S)

Given the set of public system parameter psp, a keyword w, a receiver's identity ID_U, public key $PK_U=(PK_{U1}, PK_{U2})$, a designated server's identity ID_S, and public key $PK_S=(PK_{S1}, PK_{S2})$, this algorithm randomly selects $r \in Z_q^*$ and computes a keyword ciphertext $C_w=(C_{w1}, C_{w2})=(rP, H_3(rH_2(w)Q_U+rQ_S))$, where $Q_U=PK_{U1}+PK_{U2}+H_1(ID_U)P_{pub}$ and $Q_S=PK_{S1}+PK_{S2}+H_1(ID_S)P_{pub}$.

Algorithm 5 Test algorithm in the designated server CLEKS scheme

Test(psp, SK_S, C_w, $T_{w'}$)

Given the set of public system parameter psp, a designated server's private key $SK_S=(SK_{S1}, SK_{S2})$, a keyword ciphertext $C_w=(C_{w1}, C_{w2})$, and a keyword trapdoor $T_{w'}$, this algorithm checks whether $C_{w2}=H_3((T_{w'}+SK_{S1}+SK_{S2})C_{w1})$. It outputs 1 if it does, and 0 otherwise.

5 Conclusions

In this study, we have presented a pairing-free CLEKS scheme and formally proved its security under the complexity assumption of the CDH problem in the random oracle model. Compared with the previous pairing-based CLEKS schemes, our scheme enjoys obvious advantages in terms of both the computation performance and the communication bandwidth. In addition, we have presented three extensions of the proposed CLEKS scheme.

The security of the existing CLEKS schemes can be obtained only in the random oracle model. In our future research, we may consider building a provably secure CLEKS scheme without random oracles. In addition, it would be interesting to devise CLEKS schemes that support authorized keyword search, as done for the PEKS scheme in Shi et al. (2014).

Compliance with ethics guidelines

Yang LU and Ji-guo LI declare that they have no conflict of interest.

References

Abdalla M, Bellare M, Catalano D, et al., 2008. Searchable encryption revisited: consistency properties, relation to anonymous IBE, and extensions. *J Cryptol*, 21(3):350-391. https://doi.org/10.1007/s00145-007-9006-6

Al-Riyami SS, Paterson KG, 2003. Certificateless public key cryptography. Proc 9th Int Conf on the Theory and Application of Cryptology and Information Security, p.452-473. https://doi.org/10.1007/978-3-540-40061-5_29

Baek J, Safavi-Naini R, Susilo W, 2008. Public key encryption with keyword search revisited. Proc 7th Int Conf on Computational Science and Its Applications, p.1249-1259. https://doi.org/10.1007/978-3-540-69839-5_96

Bellare M, Rogaway P, 1993. Random oracles are practical: a paradigm for designing efficient protocols. Proc 1st ACM Conf on Computer and Communications Security, p.62-73. https://doi.org/10.1145/168588.168596

Boneh D, di Crescenzo G, Ostrovsky R, et al., 2004. Public key encryption with keyword search. Proc Int Conf on the Theory and Applications of Cryptographic Techniques, p.506-522. https://doi.org/10.1007/978-3-540-24676-3_30

Byun JW, Lee DH, Lim J, 2006. Efficient conjunctive keyword search on encrypted data storage system. Proc 3rd European PKI Workshop, p.184-196. https://doi.org/10.1007/11774716_15

Canetti R, Goldreich O, Halev S, 2004. The random Oracle methodology, revisited. *J ACM*, 51(4):557-594. https://doi.org/10.1145/1008731.1008734

Dong QX, Guan Z, Wu L, et al., 2013. Fuzzy keyword search over encrypted data in the public key setting. Proc 14th Int Conf on Web-Age Information Management, p.729-740. https://doi.org/10.1007/978-3-642-38562-9_74

Fang LM, Susilo W, Ge P, et al., 2009. A secure channel free public key encryption with keyword search scheme without random oracle. Proc 8th Int Conf on Cryptology and Network Security, p.248-258. https://doi.org/10.1007/978-3-642-10433-6_16

Fu ZJ, Sun XM, Liu Q, et al., 2015. Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. *IEICE Trans Commun*, E98.B(1):190-200. https://doi.org/10.1587/transcom.E98.B.190

Fu ZJ, Ren K, Shu JG, et al., 2016. Enabling personalized search over encrypted outsourced data with efficiency improvement. *IEEE Trans Parall Distrib Syst*, 27(9): 2546-2559. https://doi.org/10.1109/TPDS.2015.2506573

Golle P, Staddon J, Waters B, 2004. Secure conjunctive

keyword search over encrypted data. Proc 2nd Int Conf on Applied Cryptography and Network Security, p.31-45. https://doi.org/10.1007/978-3-540-24852-1_3

Guo LF, Yau WC, 2015. Efficient secure-channel free public key encryption with keyword search for EMRs in cloud storage. *J Med Syst*, 39(2):1-11. https://doi.org/10.1007/s10916-014-0178-y

Hu CY, Liu PT, 2012. An enhanced searchable public key encryption scheme with a designated tester and its extensions. *J Comput*, 7(3):706-715.

Hwang YH, Lee PJ, 2007. Public key encryption with conjunctive keyword search and its extension to a multi-user system. Proc 1st Int Conf on Pairing-Based Cryptography, p.2-22. https://doi.org/10.1007/978-3-540-73489-5_2

Islam SH, Obaidat MS, Rajeev V, et al., 2017. Design of a certificateless designated server based searchable public key encryption scheme. Proc 3rd Int Conf on Mathematics and Computing, p.3-15. https://doi.org/10.1007/978-981-10-4642-1_1

Li JG, Lin XN, Zhang YC, et al., 2017a. KSF-OABE: outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Trans Serv Comput*, 10(5):715-725. https://doi.org/10.1109/TSC.2016.2542813

Li JG, Shi YR, Zhang YC, 2017b. Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage. *Int J Commun Syst*, 30(1):1-13. https://doi.org/10.1002/dac.2942

Liu JN, Lai JZ, Huang XY, 2017. Dual trapdoor identity-based encryption with keyword search. *Soft Comput*, 21(10): 2599-2607. https://doi.org/10.1007/s00500-015-1960-6

Lu Y, Wang G, Li JG, et al., 2017. Efficient designated server identity-based encryption with conjunctive keyword search. *Ann Telecommun*, 72(5-6):359-370. https://doi.org/10.1007/s12243-017-0574-7

Lv ZQ, Hong C, Zhang M, et al., 2014. Expressive and secure searchable encryption in the public key setting. Proc 17th Int Conf on Information Security, p.364-376. https://doi.org/10.1007/978-3-319-13257-0_21

MIRACL Ltd., 2012. MIRACL Cryptographic SDK: Multi-precision Integer and Rational Arithmetic Cryptographic Library. https://github.com/miracl/MIRACL

Park DJ, Kim K, Lee PJ, 2004. Public key encryption with conjunctive field keyword search. Proc 5th Int Workshop on Information Security Applications, p.73-86. https://doi.org/10.1007/978-3-540-31815-6_7

Peng YG, Cui JT, Peng CG, et al., 2014. Certificateless public key encryption with keyword search. *China Commun*, 11(11):100-113. https://doi.org/10.1109/CC.2014.7004528

Rhee HS, Park JH, Susilo W, et al., 2010. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J Syst Softw*, 83(5):763-771. https://doi.org/10.1016/j.jss.2009.11.726

Shamir A, 1984. Identity-based cryptosystems and signature schemes. Proc 1st Workshop on the Theory and Application of Cryptographic Techniques, p.47-53. https://doi.org/10.1007/3-540-39568-7_5

Shao ZY, Yang B, 2015. On security against the server in designated tester public key encryption with keyword search. *Inform Process Lett*, 115(12):957-961. https://doi.org/10.1016/j.ipl.2015.07.006

Shi J, Lai JZ, Li YJ, et al., 2014. Authorized keyword search on encrypted data. Proc 1st European Symp on Research in Computer Security, p.419-435. https://doi.org/10.1007/978-3-319-11203-9_24

Siad A, 2012. Anonymous identity-based encryption with distributed private-key generator and searchable encryption. Proc 5th Int Conf on New Technologies, Mobility and Security, p.1-8. https://doi.org/10.1109/NTMS.2012.6208695

Song DX, Wagner D, Perrig A, 2000. Practical techniques for searches on encrypted data. Proc IEEE Symp on Security and Privacy, p.44-55. https://doi.org/10.1109/SECPRI.2000.848445

Standards for Efficient Cryptography Group, 2000. SEC 2: Recommended Elliptic Curve Domain Parameters, Version 1.0. http://www.secg.org/SEC2-Ver-1.0.pdf

Sun WH, Yu SC, Lou WJ, et al., 2016. Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. *IEEE Trans Parall Distrib Syst*, 27(4):1187-1198. https://doi.org/10.1109/TPDS.2014.2355202

Tian XX, Wang Y, 2008. ID-based encryption with keyword search scheme from bilinear pairings. Proc 4th Int Conf on Wireless Communications, Networking and Mobile Computing, p.1-4. https://doi.org/10.1109/WiCom.2008.2916

Tomida K, Doi H, Mohri M, et al., 2015. Ciphertext divided anonymous HIBE and its transformation to identity-based encryption with keyword search. *J Inform Process*, 23(5): 562-569. https://doi.org/10.2197/ipsjjip.23.562

Wang XF, Mu Y, Chen RM, et al., 2016. Secure channel free ID-based searchable encryption for peer-to-peer group. *J Comput Sci Technol*, 31(5):1012-1027. https://doi.org/10.1007/s11390-016-1676-9

Wu TY, Tsai TT, Tseng YM, 2014. Efficient searchable ID-based encryption with a designated server. *Ann Telecomm*, 69(7-8):391-402. https://doi.org/10.1007/s12243-013-0398-z

Zhang B, Zhang FG, 2011. An efficient public key encryption with conjunctive-subset keywords search. *J Netw Comput Appl*, 34(1):262-267. https://doi.org/10.1016/j.jnca.2010.07.007

Zheng QJ, Xu SH, Ateniese G, 2014. VABKS: verifiable attribute-based keyword search over outsourced encrypted data. Proc IEEE INFOCOM, p.522-530. https://doi.org/10.1109/INFOCOM.2014.6847976

Zheng QJ, Li XX, Azgin A, 2015. CLKS: certificateless keyword search on encrypted data. Proc 9th Int Conf on Network and System Security, p.239-253. https://doi.org/10.1007/978-3-319-25645-0_1