Journal of Information Security and Applications 55 (2020) 102652

Contents lists available at ScienceDirect

Journal of Information Security and Applications

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

Multi-user certificateless public key encryption with conjunctive keyword search for cloud-based telemedicine

Mimi Ma [a,b,c,*], Shuqin Fan [a], Dengguo Feng [a]

[a] State Key Laboratory of Cryptology, Beijing 100878, China
[b] Key Laboratory of Grain Information Processing and Control (Henan University of Technology), Ministry of Education, Zhengzhou 450001, China
[c] College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China

ARTICLE INFO

Keywords:
Certificateless
Searchable public key encryption
Cloud-based telemedicine
Privacy

ABSTRACT

With the development of communication and information technologies, the telemedicine system has infiltrated many aspects of medicine field. It allows doctors to simultaneously diagnose patients in different areas, which provides great convenience to people. However, the increasing medical data brings serious challenges to people, such as data storage and processing. To reduce data management costs and enjoy convenient services, more and more individuals and medical institutions prefer to store data in the cloud. Recently, many public key encryption with keyword search (PEKS) schemes have been designed to address the security and privacy of the outsourced data. However, most of the existing schemes support only single-keyword search, and cannot suitable for multi-user environment. To resolve these issues, we construct a certificateless public key encryption with conjunctive keyword search scheme for multi-user system (mCLPECK), in which the same data only needs to be encrypted once and can be searched by multi-receiver. At the same time, the mCLPECK scheme supports multi-keyword search. Furthermore, we show the proposed scheme can resist chosen keyword attacks, and has lower storage and computation overhead compared to other related schemes.

1. Introduction

Telemedicine is a new medical model that uses advanced communication and information technologies (i.e., computers, mobile devices) to realize the diagnosis, treatment, medicine education and other medical functions between experts and patients at a distance [1]. At present, the telemedicine technology has developed from early telephone remote diagnosis to real-time voice and video communication using the Internet anytime and anywhere, providing a broad prospect for the development of telemedicine. As such, it breaks the regional restrictions, not only can provide medical resources for remote areas, but also can improve medical service level in big cities [2]. Furthermore, it can reduce patients' waiting time, avoid delayed diagnosis and misdiagnosis, and reduce medical expenses.

Along with the rapid development of telemedicine, it also faces many challenges, such as the storage issue of the huge electronic health records (EHRs) and the limited computing power of mobile medical monitoring devices. To address these issues, cloud-based telemedicine is emerging [3], which provides more affordable and efficient health services. Integrated with the powerful data storage capacity and the features of dynamic, scalable and pay-as-you-go service of cloud computing [4–6], cloud-based telemedicine not only provides users with convenient and high-quality services, but also reduces the burden of equipment maintenance. Fig. 1 shows the architecture for cloud-based telemedicine.

The cloud-based telemedicine brings great benefits to people, however, it also faces many serious threats especially the security and privacy issues. In a cloud-based telemedicine system, the EHRs data is outsourced to cloud server, and the users (i.e., doctors and patients) will lose physical control of these data. However, the server is not entirely trusted and may steal data for illegal commercial use. Therefore, ensuring data security and privacy is considered as a critical requirement [7,8]. To enhance data security, data is usually stored on the server in ciphertext state. However, the ciphertext data will change the original structure, so the existing search algorithm for plaintext will no longer work for ciphertext.

The technology of searchable encryption (SE) can effectively realize the function of searching on ciphertext data without revealing any information of plaintext data. Song et al. [9] firstly proposed a SE scheme based on symmetric cryptography (SSE). Later, many SSE schemes have been proposed in an effort to balance security and efficiency [10–12]. The SSE algorithm has its own inherent advantages, such as fast operation and high performance. However, due to the symmetric nature of

* Corresponding author at: College of Information Science and Engineering, Henan University of Technology, Zhengzhou 450001, China.
E-mail address: mamimi421@126.com (M. Ma).

https://doi.org/10.1016/j.jisa.2020.102652

Available online 16 October 2020

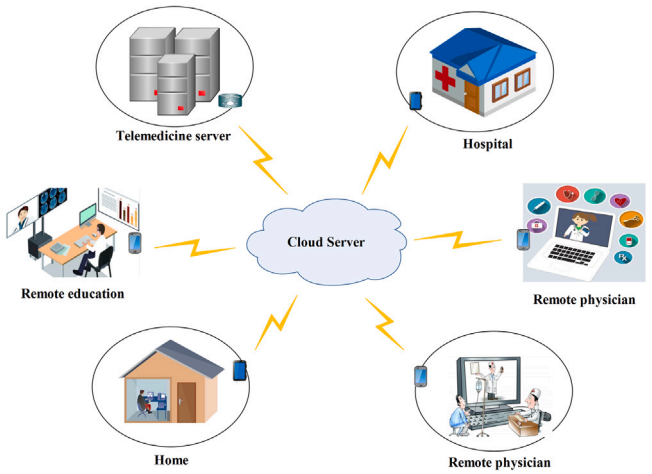

Fig. 1. The architecture for cloud-based telemedicine.

the secret key in SSE, the communicating parties must share a common secret key in advance, and each key can only used once for security. Therefore, the SSE schemes suffer from cumbersome key management issues, and are only suitable for single-user model, which limits its practical application. The reality is more of multi-user scenarios. For example, in a cloud-based telemedicine system, a patient may want to be diagnosed by more than one doctor, so he/she needs to share his/her EHRs with multiple doctors.

Motivation: To solve above issues, Boneh et al. [13] introduced the definition of PEKS and gave a PEKS instance. However, the proposed instance is not efficient in multi-user scenario. For example, when there are multiple receivers, the data sender needs to encrypt the same data multiple times using different receiver's public key, which will consume huge computing and storage overhead. Recently, many PEKS schemes for multi-owner or multi-receiver system have been designed [14–17]. Hwang et al. [14] developed a PEKS scheme that support conjunctive keyword search, and they extended their scheme to the multi-user scenario. Wang et al. [16] proposed an attribute-based keyword search scheme to achieve flexible access control in multi-user system.

However, these schemes are plagued by cumbersome certificate management or key escrow issues. To resolve these issues, the certificateless cryptosystem is introduced [18]. Recently, many certificateless PEKS (CLPEKS) schemes have been constructed [19,20]. However, the previous CLPEKS schemes were unable to effectively support multi-keyword search and only considered the single receiver scenario, which cannot satisfy the realistic demand. Therefore, it is meaningful and challenging to develop a CLPEKS scheme that supports flexible search for multi-user system.

1.1. Research contributions

The main research contributions are summarized as follows.

- Firstly, we design a new mCLPECK scheme, which supports conjunctive keyword search and is suitable for multi-user scenarios.
- Secondly, we show mCLPECK is semantically secure under the random oracle model.
- Finally, we provide the performance evaluation, and evaluation results indicate that mCLPECK has better performance.

1.2. Organization of the paper

Section 2 and Section 3 present some related references and preliminary backgrounds, respectively. Section 4 presents the concrete construction of the mCLPECK scheme. Section 5 and Section 6 provide the security and efficiency analysis for mCLPECK. We conclude this paper in Section 7.

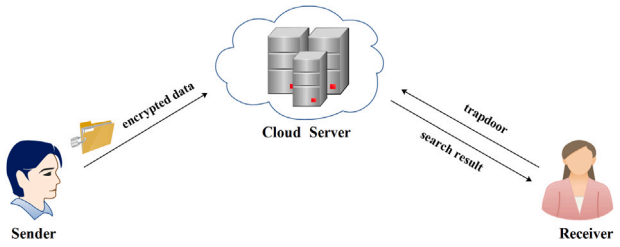

Fig. 2. The system model for PEKS.

2. Related work

To achieve data sharing between users, Boneh et al. [13] designed a PEKS scheme, which considered the e-mail system with three entities (i.e., the sender named Bob, the receiver named Alice, and the e-mail server). As shown in Fig. 2, Bob encrypts the e-mail files (i.e., $C_F = \{Enc(F_1), \ldots, Enc(F_m)\}$) and the keywords extracted from the files (i.e., $C_w = \{PEKS(w_1), \ldots, PEKS(w_n)\}$). Then, Bob uploads $\{C_F, C_w\}$ to the server. Alice generates the trapdoor $T_{w'}$ for keyword to be queried with his own private key, and transmits $T_{w'}$ to server. Upon receiving $T_{w'}$, the server tests whether the keyword contained in C_w are equal to the keyword contained in $T_{w'}$. If they are equal, the corresponding ciphertext files $C_{F'} = \{Enc(F_{I_1}), \ldots, Enc(F_{I_{m'}})\}$ are returned. Later, many PEKS schemes have been constructed [21–26].

Abdalla et al. [27] pointed out scheme [13] was computationally consistent, and they constructed an improved scheme that can achieve statistically consistent. Furthermore, they presented a transformation method between PEKS and identity-based encryption (IBE) in Ref. [27]. Wang et al. [16] integrated ciphertext-policy attribute-based encryption (CP-ABE) into PEKS and gave a concrete construction. In this construction, the receiver's private key is associated with an attribute set, and the ciphertext includes an access control policy. The receiver is authorized to perform keyword search if and only if the attribute set meets the access control policy. Miao et al. [28] constructed a PEKS scheme with privacy-preserving based on CP-ABE.

A verifiable attribute-based PEKS scheme is designed by Zheng et al. [29]. In their proposed scheme, the authorized users not only could retrieve the keywords, but also can check whether the cloud server is honestly performing this search. Liu et al. [30] designed a verifiable PEKS based on key-policy ABE (KP-ABE), in which the ciphertext contains an attribute set, and user's private key is associated with an access control policy. The user can search on the ciphertext according to keywords if the attribute set meets the access policy. Additional, the user can verify the correctness and integrity of the received retrieval results. Shao et al. [30] combined the SE technology with proxy re-encryption (PRE) technology and proposed a PRE with keyword search scheme (PRES).

Park et al. [31] constructed two PEKS schemes that support the search of conjunctive field keywords (PECKS). The trapdoor length in both schemes is constant. Liang et al. [32] designed a PECKS scheme that supports regular language retrieval over encrypted data. A verifiable PECKS scheme for dynamic data-owner environment is designed by Miao et al. [33]. And they proved their scheme could withstand keyword guessing attack (KGA) under standard model. To resist KGA attack, Xu et al. [34] presented a PEKS scheme with fuzzy keyword search, in which the trapdoor of each keyword contains two parts (a fuzzy trapdoor and an exact trapdoor). Recently, a dual-server PEKS scheme is constructed by Chen et al. [35]. The keyword search phase of this scheme needs to be completed by two servers. Huang et al. [36] developed a PEKS scheme with authentication function. However, the above schemes suffer from certificate management or key escrow issues. To resolve these problems, many certificateless schemes have been constructed [19,37–39]. Peng et al. [19] constructed a

Table 1
Notations.

Notation	Definition
1^k	Security parameter
q	Prime number
$\mathbb{G}_1, \mathbb{G}_2$	Two cyclic groups
g	A generator of $\mathbb{G}_1$.
h_0, H_1, H_2, h_3	Cryptographic hash functions
W	The keyword set
(s, g_{pub})	System master/public key pair
ID_i	The ith user identity
d_i	The ith user partial private key
(PK_i, SK_i)	The ith user public/private key pair
e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$	A bilinear pairing

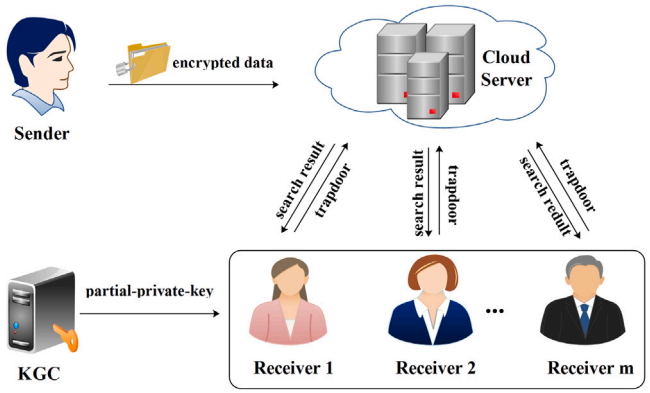

Fig. 3. A system model for mCLPECK.

certificateless PEKS (CLPEKS) to avoid cumbersome key escrow. Later, He et al. [38] designed a novel authenticated CLPEKS scheme. Wu et al. [39] developed a CLPEKS with designated verifier to prevent the KGA attacks.

3. Background

We present some notations used in this paper (see Table 1).

3.1. Preliminaries

Bilinear pairing: The mapping e: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is said a bilinear pairing if it has the following properties:

(1) *Bilinear*: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ $(\forall g_1, g_2 \in \mathbb{G}_1, a, b \in \mathbb{Z}_q^*)$.

(2) *Non-degenerate*: $e(g, g) \neq 1$.

(3) *Computable*: $e(g_1, g_2)(\forall g_1, g_2 \in \mathbb{G}_1)$ can be effectively calculated.

Decision linear Diffie–Hellman problem (DLDHP) [40]: Given six points $g_1, g_2, g_3, g_1^a, g_2^b, g_3^c \in \mathbb{G}_1$ ($a, b, c \in \mathbb{Z}_q^*$ are unknown values), to determine if $c = a + b \mod q$. We say DLDHP is intractable if the probability of any adversary successfully solving DLDHP is negligible, i.e., $Succ_{\mathbb{G}_1}^{DLDHP}(\mathcal{A}) = Pr[\mathcal{A}(g_1, g_2, g_3, g_1^a, g_2^b, g_3^c), c = a + b \mod q] \leq \epsilon$, where ϵ is negligible.

3.2. System model

Fig. 3 presents the system model of mCLPECK, which contains four types of entities (i.e., sender, receiver, cloud server and KGC). The responsibilities of each entity are as follows.

- *KGC* is a semi-trusted entity and it is responsible for producing public parameters, system master key, and the partial-private-key d_i of each receiver.
- *Sender* is a data provider, who selects a group of receivers that he/she wishes to share data with. He/She encrypts the file set F and the corresponding extracted keyword set W using the public keys of the group. Then, he/she submits ciphertext to cloud server.
- *Receiver* is the data user, who generates a trapdoor T for the keyword set he is interested in and sends T to cloud server for searching.
- *Cloud Server* is a semi-trusted entity and it is responsible for data processing (i.e., storage, computation, and search).

3.3. Threat model

As presented in the Ref. [18], there are two types of adversaries ($Type1$ adversary $\mathcal{A}_1$ and $Type2$ adversary $\mathcal{A}_2$) in a certificateless cryptosystem. $\mathcal{A}_1$ has no system master key, however, it can replace any user's public key. $\mathcal{A}_2$ has system master key, however, it cannot replace user's public key. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ can access to the following queries.

- *Extract-Partial-Private-Key query*: If the partial private key of user ID_i is queried by $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$, the challenger C executes *Extract-Partial-Private-Key* algorithm and returns a partial private key d_i.
- *Secret-Value query*: If the secret value of user ID_i is queried by $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$, the challenger C executes *Set-Secret-Value* algorithm and returns a secret value x_i.
- *Request-Public-Key query*: If $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ asks the public key of user ID_i, C performs *Set-Public-Key* algorithm and returns a public key PK_i.
- *Trapdoor query*: If $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ queries the trapdoor for a keyword set Q, C returns the trapdoor T.

Furthermore, $\mathcal{A}_1$ can access to *Replace-Public-Key query*, i.e., $\mathcal{A}_1$ can select a forged value to replace any user's real public key.

3.4. Security model

The security model for mCLPECK is defined by the following game.

Game. This game is an interaction between an adversary $\mathcal{A} \in \{\mathcal{A}_1, \mathcal{A}_2\}$ and a challenger C, as described below.

- *Setup*: C inputs the security parameter 1^k and performs *Setup* algorithm to produce public parameters $\mathcal{GP}$ and master key s. If $\mathcal{A} = \mathcal{A}_2$, C sends $\{s, \mathcal{GP}\}$ to $\mathcal{A}$; Otherwise ($\mathcal{A} = \mathcal{A}_1$), C sends $\mathcal{GP}$ to $\mathcal{A}$.
- *Queries*: $\mathcal{A}$ can perform a series of queries, including *Extract-Partial-Private-Key*, *Secret-Value*, *Request-Public-Key* and *Trapdoor* queries, which are described as Section 3.3. In addition, $\mathcal{A}_1$ can perform *Replace-Public-Key* query.
- *Challenge*: After *Queries*, $\mathcal{A}$ selects a keyword set W^* as the challenging target. C selects a keyword set R randomly and assumes $W_0 = W^*$, $W_1 = R$ (W_0 and W_1 should not be queried for *Trapdoor query*). Then, C generates a $b \in \{0, 1\}$ randomly, produces ciphertext C_{W_b}, and returns $\{C_{W_b}, W_0, W_1\}$.
- *More-Trapdoor query*: $\mathcal{A}$ can continue to query the trapdoor for keyword set W' ($W' \neq W_0, W_1$), and C outputs a corresponding value.
- *Guess*: $\mathcal{A}$ returns $b' \in \{0, 1\}$. We say $\mathcal{A}$ wins above game if $b' = b$.

Suppose $Succ_{\mathcal{A}}(1^k) = \left| Pr\left[b = b'\right] - 1/2 \right|$ denotes the probability of $\mathcal{A}$ winning the above game.

Definition 1 (*IND-CKA Secure*)**.** We say mCLPECK is semantically secure under the adaptive chosen keyword attacks if $Succ_{\mathcal{A}}(1^k)$ is negligible.

4. The construction of mCLPECK

This section presents the detail construction of mCLPECK, which includes seven algorithms.

- **Setup**(1^k): KGC inputs a security parameter 1^k and chooses a bilinear pairing e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let g be a generator of $\mathbb{G}_1$. KGC selects four different collision–resistance hash functions: $h_0 : \{0,1\}^* \times \mathbb{G}_1 \to \mathbb{Z}_q^*$, $H_1 : \{0,1\}^* \to \mathbb{G}_1$, $H_2 : \{0,1\}^* \to \mathbb{G}_1$, $h_3 : \{0,1\}^* \times \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{Z}_q^*$. KGC selects $s \in \mathbb{Z}_q^*$ randomly, and computes $g_{pub} = g^s$. KGC keeps s secretly, and publishes the public parameters

  $$\mathcal{GP} = \{\mathbb{G}_1, \mathbb{G}_2, q, g, e, g_{pub}, h_0, H_1, H_2, h_3\}.$$

- **Extract-Partial-Private-Key**($\mathcal{GP}, s, ID_1, \dots, ID_m$): KGC takes $\mathcal{GP}$, the master key s, and the receivers' identities $(ID_1, ID_2, \dots, ID_m)$ as input. KGC chooses m random numbers $s_1, s_2, \dots, s_m$ from $\mathbb{Z}_q^*$, and computes $t_i = g^{s_i}, d_i = s_i + s\alpha_i \pmod q$, where $\alpha_i = h_0(ID_i, t_i)$. Finally, KGC returns d_i and t_i to the corresponding ith receiver.
- **Set-Secret-Value**($\mathcal{GP}$): The ith receiver randomly chooses $x_i \in \mathbb{Z}_q^*$, and sets $SK_i = (x_i, d_i)$ as its own private key.
- **Set-Public-Key**($\mathcal{GP}, x_1, \dots, x_m$): The ith receiver computes $y_i = g^{x_i} (i = 1, 2, \dots, m)$, and sets $PK_i = (y_i, t_i)$ as its public key.
- **mCLPECK**($PK_1, \dots, PK_m, W$): Let the set of all keywords to be encrypted be $W = \{w_1, w_2, \dots, w_n\}$. The sender selects two values $r, r' \in Z_q^*$ randomly, and calculates $\beta_j = h_3(ID_j, g_{pub}, y_j, t_j)$,

  $$A = g^r, B_j = (y_j^{\beta_j} t_j g_{pub}^{\alpha_j})^{r'}, C_i = h_i^r f_i^{r'},$$

  where $h_i = H_1(w_i)$, $f_i = H_2(w_i)$, $1 \le i \le n$ and $1 \le j \le m$. The keywords' ciphertext is $C = (A, B_1, B_2, \dots, B_m, C_1, C_2, \dots, C_n)$.
- **Trapdoor**(SK_j, Q): Let the keyword set that the jth receiver wants to search denoted as $Q = \{I_1, I_2, \dots, I_l, w_{I_1}, w_{I_2}, \dots, w_{I_l}\}$. The receiver randomly selects $t \in \mathbb{Z}_q^*$ and calculates $\beta_j = h_3(ID_j, g_{pub}, y_j, t_j)$,

  $$T_{j,1} = g^t, T_{j,2} = (h_{I_1} h_{I_2} \cdots h_{I_l})^t, T_{j,3} = (f_{I_1} f_{I_2} \cdots f_{I_l})^{t/(\beta_j x_j + d_j)}.$$

  The trapdoor of the keyword set Q is $T_j = (T_{j,1}, T_{j,2}, T_{j,3}, I_1, I_2, \dots, I_l)$.
- **Test**(T_j, C, PK_j): The cloud server verifies

  $$e(T_{j,1}, \prod_{i=1}^{l} C_{I_i}) = e(A, T_{j,2}) \cdot e(B_j, T_{j,3}).$$

  If it is true, outputs 1, and 0 otherwise.

Correctness.

$$e(T_{j,1}, \prod_{i=1}^{l} C_{I_i}) = e(g^t, \prod_{i=1}^{l} h_{I_i}^r f_{I_i}^{r'})$$
$$= e(g^t, \prod_{i=1}^{l} h_{I_i}^r) \cdot e(g^t, \prod_{i=1}^{l} f_{I_i}^{r'}) = e(g, \prod_{i=1}^{l} h_{I_i})^{rt} \cdot e(g, \prod_{i=1}^{l} f_{I_i})^{r't},$$

$$e(A, T_{j,2}) \cdot e(B_j, T_{j,3})$$
$$= e(g^r, \prod_{i=1}^{l} h_{I_i}^t) \cdot e((y_j^{\beta_j} t_j g_{pub}^{\alpha_j})^{r'}, \prod_{i=1}^{l} f_{I_i}^{\frac{t}{\beta_j x_j + d_j}})$$
$$= e(g, \prod_{i=1}^{l} h_{I_i})^{rt} \cdot e(g^{r'(\beta_j x_j + d_j)}, \prod_{i=1}^{l} f_{I_i}^{\frac{t}{\beta_j x_j + d_j}})$$
$$= e(g, \prod_{i=1}^{l} h_{I_i})^{rt} \cdot e(g, \prod_{i=1}^{l} f_{I_i})^{r't}.$$

Thus, we have

$$e(T_{j,1}, \prod_{i=1}^{l} C_{I_i}) = e(A, T_{j,2}) \cdot e(B_j, T_{j,3}).$$

Extension. The proposed mCLPECK scheme can be extended to support user's addition and revocation by sacrificing a little storage cost of the sender. The sender needs to store the random number r' selected during the encryption phase. If a new receiver ID_{new} is added to the group, then the sender only needs to compute $B_{new} = (y_{new}^{\beta_{new}} t_{new} g_{pub}^{\alpha_{new}})^{r'}$ and add B_{new} to the ciphertext C. If a receiver ID_i is revoked from the group, then the sender only needs to delete B_i from the ciphertext C directly. If B_i is deleted, then the receiver ID_i will not be able to query the data outsourced on cloud server. This feature increases the scalability of mCLPECK and makes it more practical.

5. Security analysis

In this section, we present mCLPECK is semantically secure under the security model defined in Section 3.4.

Theorem 1. *Suppose DLDHP is intractable, then mCLPECK is semantically secure under the chosen keyword attacks.*

Theorem 1 can be derived from Lemmas 1 and 2.

Lemma 1. *There is an algorithm C that can break the DLDHP assumption if $\mathcal{A}_1$ can break the mCLPECK scheme.*

Proof. Let the maximum number of queries for the trapdoor is q_t. The next, we will present that if $\mathcal{A}_1$ can break the scheme in Section 4 with the probability ϵ, then C can solve DLDHP with the probability $\epsilon' \ge \frac{\epsilon}{4nq_t}$.

Setup: Given an instance of DLDHP assumption, i.e., $\{g_1, g_2, g_3, v_1 = g_1^a, v_2 = g_2^b, v_3 = g_3^{a+b} \text{ or } z\}$. C runs Setup algorithm. Let $g_{pub} = v_2$, $g = g_1$. Assume $g_2 = g_1^\alpha$. Then, C sends $\mathcal{GP}$ to $\mathcal{A}_1$. In addition, C chooses a number $\eta \in \mathbb{Z}_q^*$ randomly and keeps η secret.

$\mathcal{A}_1$ is allowed to query the following hash oracles, and C returns the corresponding value. Suppose C stores four hash lists, expressed as h_0^{list}, H_1^{list}, H_2^{list}, h_3^{list}.

h_0 query: Once receiving the h_0 query about (ID_i, t_i), C executes the following operations:

(1) If $\langle ID_i, t_i, \alpha_i \rangle$ exists in h_0^{list}, C outputs α_i directly.

(2) Otherwise, C selects $\alpha_i \in \mathbb{Z}_q^*$ randomly, adds $\langle ID_i, t_i, \alpha_i \rangle$ to h_0^{list} and outputs α_i.

H_1, H_2 queries: Once receiving H_1 and H_2 queries for w_i, C does the following steps:

(1) If $\langle w_i, c_i, h_i, d_i \rangle$ already exists in H_1^{list} and the tuple $\langle w_i, c_i, f_i, e_i \rangle$ in H_2^{list}, then C outputs h_i as the response of H_1 query, and outputs f_i as the response of H_2 query.

(2) Otherwise, C produces $c_i \in \{0, 1\}$ at random such that $Pr[c_i = 1] = 1/(nq_t)$. If $c_i = 0$, C chooses $d_i \in \mathbb{Z}_q^*$ and $e_i \in \mathbb{Z}_q^*$ randomly, and computes $h_i = g_1^{d_i}$, $f_i = g_2^{e_i}$. Otherwise, C generates $d_i \in \mathbb{Z}_q^*$ randomly, and calculates $h_i = g_3^{d_i}$, $f_i = g_3^{e_i}$, where $e_i = d_i/\eta$. Then, C outputs $\{h_i, f_i\}$, adds $\langle w_i, c_i, h_i, d_i \rangle$ to H_1^{list}, and adds $\langle w_i, c_i, f_i, e_i \rangle$ to H_2^{list}.

h_3 query: Upon receiving $\mathcal{A}_1$'s query for $(ID_i, g_{pub}, y_i, t_i, \beta_i)$, C runs the operations as below:

(1) If $\langle ID_i, g_{pub}, y_i, t_i, \beta_i \rangle$ exists in h_3^{list}, C outputs β_i directly.

(2) Otherwise, C randomly selects $\beta_i \in \mathbb{Z}_q^*$, adds $\langle ID_i, g_{pub}, y_i, t_i, \beta_i \rangle$ to h_3^{list} and returns β_i.

Extract-Partial-Private-Key query: A list L_1^{list} with tuples $\langle ID_i, t_i, s_i, d_i \rangle$ is stored by C. Upon receiving $\mathcal{A}_1$'s query for ID_i, C executes the following operations:

(1) If $\langle ID_i, t_i, s_i, d_i \rangle$ exists in L_1^{list}, C returns (t_i, d_i) to $\mathcal{A}_1$.

(2) Otherwise, C randomly selects $\alpha_i, d_i, s_i \in \mathbb{Z}_q^*$, computes $t_i = g_2^{d_i} g_{pub}^{-\alpha_i}$. C adds $\langle ID_i, t_i, \alpha_i \rangle$ to h_0^{list}, adds $\langle ID_i, t_i, s_i, d_i \rangle$ to L_1^{list} and returns (t_i, d_i) to $\mathcal{A}_1$.

Request-Public-Key query: A list L_2^{list} with tuples $\langle ID_i, y_i, t_i \rangle$ is stored by C. Upon receiving the query for ID_i, C performs as below:

(1) If L_2^{list} already concludes $\langle ID_i, y_i, t_i \rangle$, C returns (y_i, t_i) to $\mathcal{A}_1$.

(2) Otherwise, if $\langle ID_i, t_i, s_i, d_i \rangle$ exists in L_1^{list}, C randomly selects $\pi_i \in \mathbb{Z}_q^*$, calculates $y_i = g_2^{\pi_i}$, adds $\langle ID_i, y_i, t_i \rangle$ to L_2^{list} and returns (y_i, t_i)

to $\mathcal{A}_1$. Otherwise, C firstly queries *Extract-Partial-Private-Key* for ID_i. Then, C selects $\pi_i \in \mathbb{Z}_q^*$ randomly, computes $y_i = g_2^{\pi_i}$.

Replace-Public-Key query: When $\mathcal{A}_1$ asks this query for $\langle ID_i, PK_i' \rangle$, then C sets $PK_i \leftarrow PK_i'$.

Secret-Value query: C stores a secret value list with the tuples $\langle ID_i, \pi_i \rangle$, named L_3^{list}, and sets the corresponding public key as $y_i = g_2^{\pi_i}$. If $\mathcal{A}_1$ queries the secret value for ID_i, C performs as follows:

(1) If L_3^{list} already concludes $\langle ID_i, \pi_i \rangle$, C returns π_i to $\mathcal{A}_1$.

(2) Otherwise, C randomly chooses $\pi_i \in \mathbb{Z}_q^*$, adds $\langle ID_i, \pi_i \rangle$ to L_3^{list} and returns π_i to $\mathcal{A}_1$.

Trapdoor query: When $\mathcal{A}_1$ makes a trapdoor query for a keyword set $Q = \{w_{I_1}, w_{I_2}, \ldots, w_{I_l}\}$ with ID_i, C responds as follows:

(1) C recoveries $\langle ID_i, t_i, \alpha_i \rangle$, $\langle w_{I_j}, c_{I_j}, h_{I_j}, d_{I_j} \rangle$, $\langle w_{I_j}, c_{I_j}, f_{I_j}, e_{I_j} \rangle$ ($1 \le j \le l$), and $\langle ID_i, y_i, t_i \rangle$, from h_0^{list}, H_1^{list}, H_2^{list} and L_2^{list}, respectively.

(2) For $1 \le j \le l$, if there exists $c_{I_j} = 1$, C aborts.

(3) Otherwise, C searches the tuples $\langle ID_i, g_{pub}, y_i, t_i, \beta_i \rangle$ and $\langle ID_i, \pi_i \rangle$ from h_3^{list} and L_3^{list} respectively, chooses $t \in \mathbb{Z}_q^*$ randomly, computes $T_{i,1} = g_1^t$, $T_{i,2} = g_1^{t(\sum_{j=1}^{l} d_{I_j})}$, $T_{i,3} = g_1^{t(\sum_{j=1}^{l} e_{I_j})/(\beta_i \pi_i + d_i)}$, and outputs $T_i = \{T_{i,1}, T_{i,2}, T_{i,3}\}$.

Challenge: $\mathcal{A}_1$ outputs $W^* = (w_{0,1}, w_{0,2}, \ldots, w_{0,n})$ as the target keyword set. C randomly chooses $R = (w_{1,1}, w_{1,2}, \ldots, w_{1,n})$, and sets $W_0 = W^*$, $W_1 = R$, where the trapdoor of W_0 and W_1 cannot be queried.

C randomly selects $b \in \{0, 1\}$, and asks H_1 and H_2 queries for all keywords $w_{b,i}$ ($1 \le i \le n$), and obtains the tuples $\langle w_{b,i}, c_{b,i}, h_{b,i}, d_{b,i} \rangle$ and $\langle w_{b,i}, c_{b,i}, f_{b,i}, e_{b,i} \rangle$. If all $c_{b,i}$'s are not equal 1 ($1 \le i \le n$), then C terminates this simulation. Otherwise, C calculates $A = v_1$, $B_j = v_2^{(\beta_j \pi_j + d_j)\eta}$, and $C_{b,i} = v_1^{d_{b,i}} v_2^{e_{b,i}\eta}$ (in case that $c_{b,i} = 0$) or $v_3^{d_{b,i}}$ (in case that $c_{b,i} = 1$). Then, C sends W_0, W_1 and the ciphertext $C_b = (A, B_1, \ldots, B_m, C_{b,1}, \ldots, C_{b,n})$ to $\mathcal{A}_1$.

More-Trapdoor query: $\mathcal{A}_1$ could continue to make keyword W_i's trapdoor queries, where $W_i \notin \{W_0, W_1\}$. C responds as above.

Guess: $\mathcal{A}_1$ returns $b' \in \{0, 1\}$. C returns 1 if $b = b'$ implying $v_3 = g_3^{a+b}$. Otherwise, C returns 0 implying $v_3 = z$.

If $v_3 = g_3^{a+b}$, then we can know that the challenge ciphertext is valid since

$$A^* = v_1 = g_1^a = g^a$$

$$B_j^* = v_2^{(\beta_j \pi_j + d_j)\eta} = g_2^{(\beta_j \pi_j + d_j) b\eta} = (y_j^{\beta_j} t_j g_{pub}^{\alpha_j})^{b\eta},$$

and if $c_{b,i} = 0$,

$$C_{b,i} = v_1^{d_{b,i}} v_2^{e_{b,i}\eta} = g_1^{d_{b,i} a} g_2^{e_{b,i} b\eta} = h_{b,i}^a f_{b,i}^{b\eta},$$

otherwise,

$$C_{b,i} = v_3^{d_{b,i}} = g_3^{(a+b)d_{b,i}} = g_3^{d_{b,i} a} (g_3^{d_{b,i}/\eta})^{b\eta} = h_{b,i}^a f_{b,i}^{b\eta}. \quad \square$$

Analysis. Assume C solves DLDHP with probability ϵ'. we first give the definition of three events.

E_1: The event that C does not abort in trapdoor query phase.

E_2: The event that C does not abort in challenge phase.

E_3: The event that C does not ask trapdoor queries for W_0 and W_1.

We have

$$Pr[E_1] = \left(1 - \frac{1}{nq_t}\right)^{nq_t} \ge \frac{1}{4},$$

$$Pr[E_2|E_1] = 1 - \left(1 - \frac{1}{nq_t}\right)^{n} \ge 1 - \left(1 - \frac{1}{nq_t}\right) \ge \frac{1}{nq_t}$$

Thus, $Pr[E_1 \wedge E_2] = Pr[E_1] Pr[E_2|E_1] \ge \frac{1}{4nq_t}$.

$\mathcal{A}_1$ can break the proposed scheme with ϵ, so $\left|Pr[b' = b] - 1/2\right| \ge \epsilon$. Since

$$\begin{aligned} &Pr[b' = b] \\ &= Pr[b' = b|\neg E_3] Pr[\neg E_3] + Pr[b' = b|E_3] Pr[E_3] \\ &\le Pr[\neg E_3] + Pr[b' = b|E_3] Pr[E_3] \\ &= Pr[\neg E_3] + \frac{1}{2} Pr[E_3] = \frac{1}{2} + \frac{1}{2} Pr[\neg E_3] \end{aligned}$$

and $Pr[b' = b] \ge Pr[E_3] Pr[b' = b|E_3] = \frac{1}{2} - \frac{1}{2} Pr[\neg E_3]$, then

$$\frac{1}{2} Pr[\neg E_3] \ge \left| Pr[b' = b] - \frac{1}{2} \right| \ge \epsilon.$$

Hence, $Pr[\neg E_3] \ge 2\epsilon$, and

$$\epsilon' \ge \frac{1}{2} Pr[\neg E_3] Pr[E_1 \wedge E_2] = \frac{1}{2} 2\epsilon \cdot \frac{1}{4nq_t} = \frac{\epsilon}{4nq_t}.$$

Lemma 2. *There is an algorithm C that can solve the DLDHP assumption if $\mathcal{A}_2$ can break the mCLPECK scheme.*

Proof. Assume q_t represents the maximum number of trapdoor queries. We will show that if $\mathcal{A}_2$ can break mCLPECK with probability ϵ, then C can solve DLDHP assumption with probability $\epsilon' \ge \frac{\epsilon}{4nq_t}$.

Setup: Given an instance of the DLDHP assumption, i.e., $\{g_1, g_2, g_3, v_1 = g_1^a, v_2 = g_2^b, v_3 = g_3^{a+b} \text{ or } z\}$. C executes *Setup* algorithm to produce public parameters $\mathcal{GP}$. C randomly selects $s \in \mathbb{Z}_q^*$. Let $g = g_1$, $g_{pub} = g_2^s$, $g_2 = g_1^\alpha$. C sends $\mathcal{GP}$ and s to $\mathcal{A}_2$. In addition, C selects $\eta \in \mathbb{Z}_q^*$ randomly and keeps it secret.

$\mathcal{A}_2$ is allowed to query all of the following hash oracles, and C outputs the corresponding values. To answer $\mathcal{A}_2$'s queries, C stores four hash lists, i.e., h_0^{list}, H_1^{list}, H_2^{list}, and h_3^{list}.

h_0 *query*: Upon receiving $\mathcal{A}_2$'s query for (ID_i, t_i), C runs the following operations:

(1) If $\langle ID_i, t_i, \alpha_i \rangle$ exists in h_0^{list}, C outputs α_i.

(2) Otherwise, C selects a value $\alpha_i \in \mathbb{Z}_q^*$ randomly, adds $\langle ID_i, t_i, \alpha_i \rangle$ to h_0^{list} and outputs α_i.

H_1, H_2 *queries*: Once receiving H_1 and H_2 queries for w_i, C does the following steps:

(1) If $\langle w_i, c_i, h_i, d_i \rangle$ and $\langle w_i, c_i, f_i, e_i \rangle$ already concluded in H_1^{list} and H_2^{list}, then C outputs h_i as the response of H_1 query, and outputs f_i as the response of H_2 query.

(2) Otherwise, C produces $c_i \in \{0, 1\}$ randomly such that $Pr[c_i = 1] = 1/(nq_t)$. If $c_i = 0$, C randomly chooses $d_i, e_i \in \mathbb{Z}_q^*$, and computes $h_i = g_1^{d_i}$, $f_i = g_2^{e_i}$. Otherwise, C selects $d_i \in \mathbb{Z}_q^*$ randomly, and calculates $h_i = g_3^{d_i}$, $f_i = g_3^{e_i}$, where $e_i = d_i/\eta$. Then, C outputs $\{h_i, f_i\}$, adds $\langle w_i, c_i, h_i, d_i \rangle$ to H_1^{list}, and adds $\langle w_i, c_i, f_i, e_i \rangle$ to H_2^{list}.

h_3 *query*: When $\mathcal{A}_2$ asks this query for $(ID_i, g_{pub}, y_i, t_i)$, C operates as follows:

(1) If the tuple $\langle ID_i, g_{pub}, y_i, t_i, \beta_i \rangle$ already exists in h_3^{list}, C directly returns β_i.

(2) Otherwise, C randomly chooses $\beta_i \in \mathbb{Z}_q^*$, adds $\langle ID_i, g_{pub}, y_i, t_i, \beta_i \rangle$ to h_3^{list} and outputs β_i.

Extract-Partial-Private-Key query: C stores a list L_1^{list}, which maintains tuples $\langle ID_i, t_i, s_i, d_i \rangle$. Upon receiving $\mathcal{A}_2$'s query, C executes the operations as below:

(1) If $\langle ID_i, t_i, s_i, d_i \rangle$ exists in L_1^{list}, C returns (t_i, d_i) to $\mathcal{A}_2$.

(2) Otherwise, C randomly selects $\alpha_i, s_i \in \mathbb{Z}_q^*$, computes $t_i = g_2^{s_i} = g_1^{\alpha s_i}$, $d_i = \alpha(s_i + s\alpha_i)$, $h_0(ID_i, t_i) = \alpha_i$, adds $\langle ID_i, t_i, \alpha_i \rangle$ and $\langle ID_i, t_i, s_i, d_i \rangle$ to h_0^{list} and L_1^{list} respectively, and returns (t_i, d_i) to $\mathcal{A}_2$.

Request-Public-Key query: C stores a list L_2^{list} with tuples $\langle ID_i, y_i, t_i \rangle$. Upon receiving $\mathcal{A}_2$'s query for ID_i, C runs the following operations:

(1) If $\langle ID_i, y_i, t_i \rangle$ exists in L_2^{list}, C returns (y_i, t_i) to $\mathcal{A}_2$.

(2) Otherwise, C chooses two numbers $s_i, \pi_i \in \mathbb{Z}_q^*$ randomly, computers $t_i = g_2^{s_i}$, $y_i = g_2^{\pi_i}$, at this time, the ith receiver's secret value is regarded as $\alpha\pi_i$, where $g_2 = g_1^\alpha$. C adds $\langle ID_i, y_i, t_i \rangle$ to L_2^{list} and returns (y_i, t_i) to $\mathcal{A}_2$.

Trapdoor query: When $\mathcal{A}_2$ makes a trapdoor query for a keyword set $Q = \{w_{I_1}, w_{I_2}, \ldots, w_{I_l}\}$ with identity ID_i, then C responds as follows:

(1) C recovers $\langle w_{I_j}, c_{I_j}, h_{I_j}, d_{I_j} \rangle$ and $\langle w_{I_j}, c_{I_j}, f_{I_j}, e_{I_j} \rangle$ ($1 \le j \le l$) from H_1^{list} and H_2^{list} respectively.

(2) If there exists $c_{I_j} = 1$ for $1 \le j \le l$, C aborts.

(3) Otherwise, C recovers the tuples $\langle ID_i, t_i, s_i, d_i \rangle$ and $\langle ID_i, g_{pub}, y_i, t_i, \beta_i \rangle$ from L_1^{list} and h_3^{list} respectively. Then, C randomly chooses $t \in \mathbb{Z}_q^*$, and computes $T_{i,1} = g_1^t$, $T_{i,2} = g_1^{t(\sum_{j=1}^{l} d_{I_j})}$, $T_{i,3} = g_1^{t(\sum_{j=1}^{l} e_{I_j})/(\beta_i \pi_i + s_i + s\alpha_i)}$.

Challenge: $\mathcal{A}_2$ will output $W^* = (w_{0,1}, w_{0,2}, \ldots, w_{0,n})$ as the target keyword set. C randomly chooses $R = (w_{1,1}, w_{1,2}, \ldots, w_{1,n})$, and sets $W_0 = W^*$, $W_1 = R$, where both W_0 and W_1 should not be asked for trapdoor queries. C randomly chooses $b \in \{0,1\}$, and obtains the tuples $\langle w_{b,i}, c_{b,i}, h_{b,i}, d_{b,i}\rangle$ and $\langle w_{b,i}, c_{b,i}, f_{b,i}, e_{b,i}\rangle$ by asking H_1 and H_2 queries for all keywords $w_{b,i}$ $(1 \le i \le n)$. If $\forall 1 \le i \le n$, $c_{b,i} = 0$, then C terminates the simulation. Otherwise, C computes $A = v_1$, $B_j = (v_2^{\beta_j \pi_j + s_j + s\alpha_j})^\eta$,

$$C_{b,i} = \begin{cases} v_1^{d_{b,i}} v_2^{e_{b,i}\eta}, & \text{if } c_{b,i} = 0; \\ v_3^{d_{b,i}}, & \text{if } c_{b,i} = 1. \end{cases}$$

Then, C sends W_0, W_1 and the ciphertext $C_b = (A, B_1, \ldots, B_m, C_{b,1}, \ldots, C_{b,n})$ to $\mathcal{A}_2$.

More-Trapdoor query: $\mathcal{A}_2$ can continue to query trapdoor for W_i, where $W_i \neq W_0, W_1$. C responds as above.

Guess: Finally, $\mathcal{A}_2$ returns $b' \in \{0,1\}$. C returns 1 if $b = b'$ implying $v_3 = g_3^{a+b}$. Otherwise, C returns 0 implying $v_3 = z$.

If $v_3 = g_3^{a+b}$, then from the following equations, we can see that the challenge ciphertext is valid,

$$A^* = v_1 = g_1^a = g^a,$$

$$B_j^* = (v_2^{\beta_j \pi_j + s_j + s\alpha_j})^\eta = (g_2^{\beta_j \pi_j + s_j + s\alpha_j})^{b\eta} = (y_j^{\beta_j} t_j g_{pub}^{\alpha_j})^{b\eta},$$

if $c_{b,i} = 0$,

$$C_{b,i} = v_1^{d_{b,i}} v_2^{e_{b,i}\eta} = g_1^{d_{b,i}a} g_2^{e_{b,i}b\eta} = h_{b,i}^a f_{b,i}^{b\eta},$$

otherwise,

$$C_{b,i} = v_3^{d_{b,i}} = g_3^{(a+b)d_{b,i}} = g_3^{d_{b,i}a} (g_3^{d_{b,i}/\eta})^{b\eta} = h_{b,i}^a f_{b,i}^{b\eta}. \quad \square$$

Analysis. Assume that C can correctly decide the DLDHP assumption with probability ϵ'. Let E_1 denote the event that C does not abort in trapdoor query phase. Let E_2 denote the event that C does not abort in challenge phase and let E_3 denote the event that C does not ask trapdoor queries for W_0 and W_1.

We have

$$Pr[E_1] = \left(1 - \frac{1}{nq_t}\right)^{nq_t} \ge \frac{1}{4},$$

$$Pr[E_2|E_1] = \left(1 - \left(1 - \frac{1}{nq_t}\right)^n\right) \ge 1 - \left(1 - \frac{1}{nq_t}\right) \ge \frac{1}{nq_t}.$$

Thus, $Pr[E_1 \wedge E_2] = Pr[E_1]Pr[E_2|E_1] \ge \frac{1}{4nq_t}$. As the proof of Lemma 1, we have $Pr[\neg E_3] \ge 2\epsilon$. So,

$$\epsilon' \ge \frac{1}{2} Pr[\neg E_3] Pr[E_1 \wedge E_2] = \frac{1}{2} \cdot 2\epsilon \cdot \frac{1}{4nq_t} = \frac{\epsilon}{4nq_t}.$$

6. Performance analysis

We present the efficiency comparison of mCLPECK and the related PECK schemes proposed in [31]. To achieve the security level of 1024-bit RSA algorithm, a Tate pairing e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is used in the implementation, where $\mathbb{G}_1$ with order q is generated by a point on a super-singular curve $E(\mathbb{F}_p)$: $y^2 = x^3 - 3x$, p and q are 512-bit and 160-bit prime numbers respectively.

6.1. Computation cost

We first give the definitions of some basic operations, as shown in Table 2. This basic operations are implemented on a personal computer (Lenovo with Windows 10 operating system, I5-8250U 1.60 GHz processor and 8 GB memory) using MIRACL library [41].

For comparison, we assume that there are m receivers. Table 3 and Figs. 4–7 show the computational costs at each phase (i.e., key generation, encryption , trapdoor generation and test phase) of the mCLPECK scheme and the existing schemes (i.e., PECK1, PECK2) proposed in [31].

Table 2
The executing time of some basic operations (ms).

	Definition	Time
T_{bp}	A bilinear pairing operation	13.144
T_{sm}	A scalar multiplication operation in $\mathbb{G}_1$	4.800
T_h	A general hash operation	0.005
T_H	A hash-to-point operation	12.082
T_{mul}	A point multiplication operation in $\mathbb{G}_2$	0.006
T_{pa}	A point addition operation in $\mathbb{G}_1$	0.025

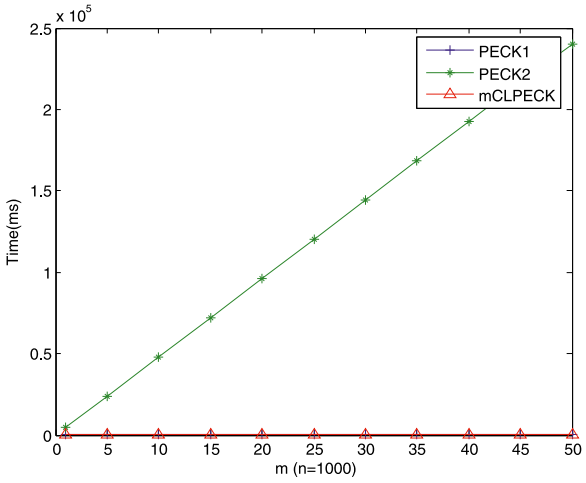

Fig. 4. KeyGen (ms).

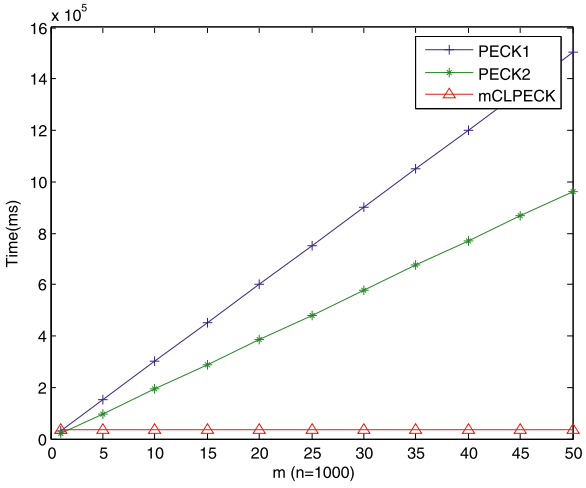

Fig. 5. Encryption (ms).

The proposed mCLPECK scheme needs to run $2m$ scalar multiplication operations, m hash operations in the key generation phase. There are $(3m + 2n + 1)$ scalar multiplication operations, $2n$ hash-to-point operations, $2m$ general hash operations and $(2m+n)$ point addition operations to encrypt n keywords. To generate the trapdoor of l keywords, mCLPECK needs to run $2l$ hash-to-point operations, one general hash operations, three scalar multiplication operations, and $2(l-1)$ point addition operations. In the test phase, there are $(l-1)$ point addition, one point multiplication, and three bilinear pairing operations.

In PECK1, there are $2m$ scalar multiplication operations in the key generation phase. To encrypt n keywords for m receivers, the number of hash-to-point and bilinear pairing operations is mn. The number of scalar multiplication operation is $m(n+2)$. To generate the trapdoor of l keywords, there are l hash-to-point operations, $(l-1)$ point addition operations, and 1 scalar multiplication operation. In the test phase,

Table 3
Comparison of computation costs (ms).

Scheme	KeyGen	Encryption
PECK1 [31]	$2mT_{sm}$	$mnT_H + m(n+2)T_{sm} + mnT_{bp}$
PECK2 [31]	$m(n+2)T_{sm} + T_{bp}$	$m(n+1)T_h + m(4n+2)T_{sm} + 2mnT_{pa}$
mCLPECK	$mT_h + 2mT_{sm}$	$2nT_H + 2mT_h + (3m+2n+1)T_{sm} + (2m+n)T_{pa}$
Scheme	**Trapdoor**	**Test**
PECK1 [31]	$lT_H + T_{sm} + (l-1)T_{pa}$	$(l-1)T_{mul} + T_{sm} + T_{bp} + T_{pa}$
PECK2 [31]	$2T_{sm} + lT_h$	$T_{mul} + T_{sm} + (2l-1)T_{pa} + 2T_{bp} + T_h$
mCLPECK	$2lT_H + T_h + 3T_{sm} + 2(l-1)T_{pa}$	$3T_{bp} + (l-1)T_{pa} + T_{mul}$

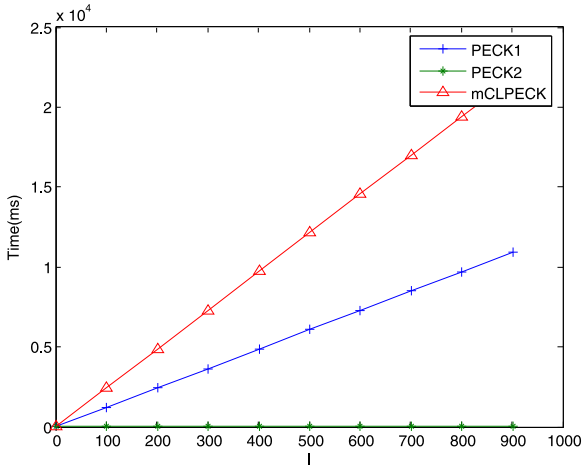

Fig. 6. Trapdoor (ms).

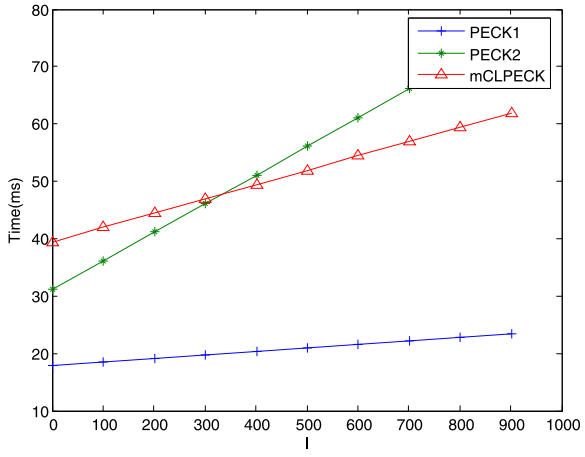

Fig. 7. Test (ms).

there are $(l-1)$ point multiplication operations, one point addition operation, one scalar multiplication operation, and one bilinear pairing operation.

In PECK2, there are $m(n+2)$ scalar multiplication operations and one bilinear pairing operation in the key generation phase. To encrypt n keywords for m receivers, there are $m(n+1)$ hash operations, $m(4n+2)$ scalar multiplication operations, and $2mn$ point addition operations. To generate the trapdoor of l keywords, there are l hash operations, and two scalar multiplication operations. In the test phase, there are one point multiplication operation, one scalar multiplication operation, $(2l-1)$ point addition operations, two bilinear pairing operations, one hash operation.

Table 3 and Figs. 4–7 illustrate the computational cost of mCLPECK is lower than that of other schemes in key generation and encryption phases. In trapdoor and test phases, although the computational cost of mCLPECK is slightly higher than that of others, the storage cost of mCLPECK is much smaller than that of PECK1 and PECK2. Thus, the computational cost of mCLPECK is acceptable.

Table 4
The communication costs (bits).

Scheme	$\|SK\|$	$\|C\|$	$\|T\|$
PECK1 [31]	$2m\|\mathbb{Z}_q^*\|$	$m(n\|\mathbb{G}_2\| + 2\|\mathbb{G}_1\|)$	$\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\|$
PECK2 [31]	$m(n+2)\|\mathbb{Z}_q^*\|$	$m((2n+1)\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\|)$	$2\|\mathbb{G}_1\| + \|\mathbb{Z}_q^*\|$
mCLPECK	$2m\|\mathbb{Z}_q^*\|$	$(m+n+1)\|\mathbb{G}_1\|$	$3\|\mathbb{G}_1\|$

6.2. Communication cost

We analysis the communication overhead of mCLPECK and the schemes (i.e., PECK1, PECK2) presented in [31]. Assume there are m receivers and n keywords. For simplicity, let us predefine a few symbols.

- $|\mathbb{Z}_q^*|$: the bit size of a number in $\mathbb{Z}_q^*$.
- $|\mathbb{G}_i|$: the bit size of a point in $\mathbb{G}_i$ $(i \in \{1,2\})$.
- $|SK|$: the size of the private key.
- $|C|$: the ciphertext size of the keyword set to be encrypted.
- $|T|$: the trapdoor size of the keyword set to be searched.

Table 4 shows the communication cost comparison between the proposed mCLPECK scheme and the schemes PECK1 and PECK2 [31].

From Table 4, we can see that the proposed mCLPECK scheme has a shorter ciphertext and private key size than the other two schemes, which will save a lot of storage overhead for cloud server and users. Although the trapdoor length in PECK1 and PECK2 is slightly shorter than that in the proposed mCLPECK scheme, they consume huge storage space, so the mCLPECK scheme is better.

7. Conclusion

The telemedicine system combines telecommunication technology, new electronic technology and computer multimedia technology to provide patients with telemedicine diagnosis and other medical services. With the gradual maturity of telemedicine technology, it has been gradually applied in many departments of medicine, such as radiology, dermatology, cardiology and neurology. However, as the wide application of telemedicine, it is inevitable to face various challenges, especially the big data storage. Cloud computing provides a powerful data processing platform for telemedicine. However, data outsourcing will face serious threats in security and privacy. To ensure data security and provide searchability, we present a new mCLPECK scheme, which supports multi-keyword search and is suitable for multi-receiver scenario. Furthermore, security analysis show that mCLPECK can resist chosen keyword attacks, and performance analysis indicate that mCLPECK has lower communication and computation costs compared to other related schemes.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (No. 61902111), the High-level talent Fund Project of Henan University of Technology, China (No. 2018BS052), the Project funded by China Postdoctoral Science Foundation (No. 2020M670223), and the Open Fund Project of Key Laboratory of Grain Information Processing and Control (Henan University of Technology, China), Ministry of Education (No. KFJJ-2016-107).

References

[1] Baker J, Stanley A. Telemedicine technology: a review of services, equipment, and other aspects. Curr Allergy Asthma Rep 2018;18(11):60.
[2] Matusitz J, Breen G-M. Telemedicine: Its effects on health communication. Health Commun 2007;21(1):73–83.
[3] Jin Z, Chen Y. Telemedicine in the cloud era: Prospects and challenges. IEEE Pervasive Comput 2015;14(1):54–61.
[4] Tang S, Li X, Huang X, Xiang Y, Xu L. Achieving simple, secure and efficient hierarchical access control in cloud computing. IEEE Trans Comput 2016;65(7):2325–31.
[5] Li X, Tang S, Xu L, Wang H, Chen J. Two-factor data access control with efficient revocation for multi-authority cloud storage systems. IEEE Access 2017;5:393–405.
[6] Chen C-M, Huang Y, Wang K-H, Kumari S, Wu M-E. A secure authenticated and key exchange scheme for fog computing. Enterpr Inf Syst 2020;1–16.
[7] Abbas A, Khan SU. A review on the state-of-the-art privacy-preserving approaches in the e-health clouds. IEEE J Biomed Health Inf 2014;18(4):1431–41.
[8] Tang S, Xu L, Liu N, Huang X, Ding J, Yang Z. Provably secure group key management approach based upon hyper-sphere. IEEE Trans Parallel Distrib Syst 2014;25(12):3253–63.
[9] Song X, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In: Security and privacy, 2000. S&P 2000. Proceedings. 2000 IEEE symposium on. IEEE; 2000, p. 44–55.
[10] Sun S-F, Yuan X, Liu JK, Steinfeld R, Sakzad A, Vo V, Nepal S. Practical backward-secure searchable encryption from symmetric puncturable encryption. In: Proceedings of the 2018 ACM SIGSAC conference on computer and communications security. ACM; 2018, p. 763–80.
[11] Soleimanian A, Khazaei S. Publicly verifiable searchable symmetric encryption based on efficient cryptographic components. Des Codes Cryptogr 2019;87(1):123–47.
[12] Ge X, Yu J, Zhang H, Hu C, Li Z, Qin Z, Hao R. Towards achieving keyword search over dynamic encrypted cloud data with symmetric-key based verification. IEEE Trans Dependable Secure Comput 2019. doi: 10.1109/TDSC.2019.2896258.
[13] Boneh D, 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.
[14] Hwang YH, Lee PJ. Public key encryption with conjunctive keyword search and its extension to multi-user system. In: International conference on pairing-based cryptography. Springer; 2007, p. 2–22.
[15] Zhang W, Lin Y, Xiao S, Wu J, Zhou S. Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing. IEEE Trans Comput 2016;65(5):1566–77.
[16] Wang C, Li W, Li Y, Xu X. A ciphertext-policy attribute-based encryption scheme supporting keyword search function. In: Cyberspace safety and security. Springer; 2013, p. 377–86.
[17] Liang K, Susilo W. Searchable attribute-based mechanism with efficient data sharing for secure cloud storage. IEEE Trans Inf Forensics Secur 2015;10(9):1981–92.
[18] Al-Riyami SS, Paterso KG. Certificateless public key cryptography. In: International conference on the theory and application of cryptology and information security. Springer; 2003, p. 452–73.
[19] Yanguo P, Jiangtao C, Changgen P, Zuobin Y. Certificateless public key encryption with keyword search. China Commun 2014;11(11):100–13.
[20] Islam SH, Obaidat MS, Rajeev V, Amin R. Design of a certificateless designated server based searchable public key encryption scheme. In: International conference on mathematics and computing. Springer; 2017, p. 3–15.
[21] Ameri MH, Delavar M, Mohajeri J, Salmasizadeh M. A key-policy attribute-based temporary keyword search scheme for secure cloud storage. IEEE Trans Cloud Comput 2018. doi: 10.1109/TCC.2018.2825983.
[22] Zhou R, Zhang X, Du X, Wang X, Yang G, Guizani M. File-centric multi-key aggregate keyword searchable encryption for industrial internet of things. IEEE Trans Ind Inf 2018;14(8):3648–58.
[23] Xu P, Tang S, Xu P, Wu Q, Hu H, Susilo W. Practical multi-keyword and boolean search over encrypted e-mail in cloud server. IEEE Trans Serv Comput 2019. doi: 10.1109/TSC.2019.2903502.
[24] Behnia R, Ozmen MO, Yavuz AA. Lattice-based public key searchable encryption from experimental perspectives. IEEE Trans Dependable Secure Comput 2018. doi: 10.1109/TDSC.2018.2867462.
[25] Zeng M, Qian H-F, Chen J, Zhang K. Forward secure public key encryption with keyword search for outsourced cloud storage. IEEE Trans Cloud Comput 2019. doi: 10.1109/TCC.2019.2944367.
[26] Chen B, Wu L, Kumar N, Choo K-KR, He D. Lightweight searchable public-key encryption with forward privacy over iiot outsourced data. IEEE Trans Emerg Top Comput 2019. http://dx.doi.org/10.1109/TETC.2019.2921113.
[27] Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Annual international cryptology conference. Springer; 2005, p. 205–22.
[28] Miao Y, Liu X, Choo K-KR, Deng RH, Li J, Li H, Ma J. Privacy-preserving attribute-based keyword search in shared multi-owner setting. IEEE Trans Dependable Secure Comput 2019. doi: 10.1109/TDSC.2019.2897675.
[29] Zheng Q, Xu S, Ateniese G. Vabks: verifiable attribute-based keyword search over outsourced encrypted data. In: IEEE INFOCOM 2014-IEEE conference on computer communications. IEEE; 2014, p. 522–30.
[30] Liu P, Wang J, Ma H, Nie H. Efficient verifiable public key encryption with keyword search based on kp-abe. In: 2014 ninth international conference on broadband and wireless computing, communication and applications. IEEE; 2014, p. 584–9.
[31] Park DJ, Kim K, Lee PJ. Public key encryption with conjunctive field keyword search. In: International workshop on information security applications. Springer; 2004, p. 73–86.
[32] Liang K, Huang X, Guo F, Liu JK. Privacy-preserving and regular language search over encrypted cloud data. IEEE Trans Inf Forensics Secur 2016;11(10):2365–76.
[33] Miao Y, Ma J, Liu X, Liu Z, Shen L, Wei F. Vmkdo: Verifiable multi-keyword search over encrypted cloud data for dynamic data-owner. Peer-to-Peer Netw Appl 2018;11(2):287–97.
[34] Xu P, Jin H, Wu Q, Wang W. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. IEEE Trans Comput 2013;62(11):2266–77.
[35] Chen R, Mu Y, Yang G, Guo F, Wang X. Dual-server public-key encryption with keyword search for secure cloud storage. IEEE Trans Inf Forensics Secur 2016;11(4):789–98.
[36] Huang Q, Li H. An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inform Sci 2017;403:1–14.
[37] He D, Kumar N, Zeadally S, Wang H. Certificateless provable data possession scheme for cloud-based smart grid data management systems. IEEE Trans Ind Inf 2018;14(3):1232–41.
[38] 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.
[39] Wu T-Y, Chen C-M, Wang K-H, Meng C, Wang EK. A provably secure certificateless public key encryption with keyword search. J Chin Inst Eng 2019;42(1):20–8.
[40] Boneh D, Boyen X, Shacham H. Short group signatures. In: Annual international cryptology conference. Springer; 2004, p. 41–55.
[41] Shamus software ltd. miracl library. 2015, http://www.shamus.ie/index.php?page=home.