Certificate-Less Searchable Encryption with a Refreshing Keyword Search

Kuma Ejeta
Addis Ababa University, kumabek4@gmail.com

Minale Ashagrie
Addis Ababa University, minale.ashagrie@aau.edu.et

Certificate-Less Searchable Encryption with a Refreshing Keyword Search

Kuma Ejeta, Minale Ashagrie
Addis Ababa University, Addis Ababa, Ethiopia
kumabek4@gmail.com
minale.ashagrie@aau.edu.et

Abstract
Public Key Encryptions with Keyword Search (PEKS) scheme had been hosted for keeping data security and privacy of outsourced data in a cloud environment. It is also used to provide search operations on encrypted data. Nevertheless, most of the existing PEKS schemes are disposed to key-escrow problems due to the private key of the target users are known by the Key Generating Center (KGC). To improve the key escrow issue in PEKS schemes, the Certificate-Less Public Key Encryptions with Keyword Search (CL-PEKS) scheme has been designed. Meanwhile, the existing CL-PEKS schemes do not consider refreshing keyword searches. Due to this, the cloud server can store search trapdoors for keywords used in the system and can launch keyword guessing attacks. In this research work, we proposed Certificate-Less Searchable Encryption with a Refreshing Keyword Search (CL-SERKS) scheme by attaching date information to the encrypted data and keyword. We demonstrated that our proposed scheme is secure against adaptively chosen keyword attacks against both types of adversaries, where one adversary is given the power to select a random public key as a replacement for the user's public key whereas another adversary is allowed to learn the system master key in the random oracle model under the Bilinear Diffie-Hellman problem assumption. We evaluated the performance of the proposed scheme in terms of both computational cost and communication cost. Experimental results show that the proposed CL-SERKS scheme has better computational cost during the key generation phase and testing phase than two related schemes. It also has lower communication costs than both related schemes.

Keywords
Refreshing keyword search, key escrow, trapdoor.

1. Introduction

A cloud is an environment that allows ubiquitous resource sharing and data access to the client efficiently and effectively while reducing the up-front infrastructure costs (Kamara & Lauter, 2010). Apart from the enormous advantages of relying upon the cloud, it also poses a serious threat to the privacy and security of the client and the data that is outsourced. To keep the security of sensitive data, cryptographic encryption mechanisms are used to encrypt the user data before outsourcing it to the cloud (Bosch, 2014). However, how to process and search on the encrypted data becomes an intractable problem (Kamara & Lauter, 2010). To solve these problems, two methods were designed. The first method requires downloading the whole ciphertext data, decrypt it locally, and then search for the preferred results in the plaintext data. This approach would be impractical for most applications, because it requires downloading a large number of files and a lot of computational cost for decryption and (Bosch, 2014).
The second method is to let the server decrypt the data, runs the query on the server-side, and sends only the results back to the target user (Bosch, 2014). In this method, the target user sends the secret key to the cloud server to decrypt the query. This lets the server learn the plaintext being queried and hence makes encryption less useful. Instead, it is suitable to support the fullest possible search functionality on the server-side, without decrypting the data which is called searchable encryption (Bosch, 2014).
Searchable encryption can be classified into symmetric and asymmetric encryption. In 2000, Song et al (Song D X, 2000) first provided a practical searchable encryption technology, which became groundbreaking in the development of searchable encryption. The scheme is based on performing a sequential scan of the document without an index. Following Song et al (Song D X, 2000), Kamara et al (Kamara S, 2012) and Wang et al (Wang G. L., 2017) proposed searchable encryption schemes based on symmetric cryptography. In symmetric searchable encryption, the encryption and decryption parties need to exchange the key beforehand. To address this limitation, public-key searchable encryption (PEKS) was first proposed by Boneh et al. (D. Boneh, 2004). The PEKS doesn't require a prior key agreement

between the data owner and target user, the data owner generates ciphertext containing both encrypted documents and encrypted keywords using the target user's public key. Then, upload the ciphertext to the cloud service provider. When the target user needs to search the ciphertext for a certain keyword, it uses the secret key to generate the search trapdoor of the keyword and sends it to the cloud server. The server then runs a test operation to select the ciphertext file containing the target keyword and returns it to the target users.

On the other hand, certificate-less searchable encryption with keyword search (CL-SEKS) scheme is a public-key cryptosystem based on identity-based encryption (Zhou, 2020). It enables a cloud service provider to get the search trapdoor to identify ciphertext containing the target keyword without decrypting the ciphertext or knowing the target keyword. The private key in certificate-less public key encryption is no longer independently generated by the Public Key Generator (PKG), which is a device or program used to generate keys, but it is jointly generated by the PKG and the target users (Al-Riyami, 2003). Since certificate-less encryption based on the identity-based public-key cryptosystem was proposed by Al-Riyami et al (Al-Riyami, 2003), the scheme overcomes the problem of key escrow in identity-based searchable encryption, in which completely trusted private key generator can know all users' private keys. This is because it supports partial private key generation by the PKG for the target users and provides to generate their secret key, which is only known by the target user. Additionally, the scheme preserves the certificate-less advantages, it overcomes certificate management problems based on PKI encryption.

Despite the certificate-less schemes provides the above-mentioned advantages, there are remaining enormous issues and challenges. To perform certificate-less searchable encryption with keyword search, the data owner encrypts both keywords and data using the target users public key and identity (Yanguo, Jiangtao, Changgen, & Zuobin, 2014). Then, it sends the encrypted document to the cloud service provider. Then, the target user generates a search trapdoor using its complete private key and sends it to the cloud service provider to conduct a keyword search. Whenever the target user sends the trapdoors, the cloud service provider can store a trapdoor and it can use the trapdoors to search in ciphertexts (Baek, Safavi-Naini, & Susilo, 2008). By using the stored trapdoors for keywords used in the system, an adversary can launch a keyword guessing attack by checking all the encrypted documents and keywords without receiving the trapdoor from the receiver (J. Li, 2017). Even though there was a proposed certificate-less-based keyword searchable encryption (Yanguo, Jiangtao, Changgen, & Zuobin, 2014), (Baek, Safavi-Naini, & Susilo, 2008), (J. Li, 2017) the trapdoors for a keyword were never refreshed. For this reason, their schemes are vulnerable to keyword guessing attacks (KGA), which allows an attacker to recover the keyword from the trapdoor (Zhou, 2020). A keyword guessing attack is a type of attack in which the attacker can correctly guess the keyword encoded in a given keyword trapdoor (Bosch, 2014). This attacker can be, the outside keyword guessing attack (OKGA), a malicious entity that has no relationship with the cloud service provider. It is also called adversary type one and represented as A1. The inside keyword guessing attack (IKGA), which is usually launched by the cloud service provider or any other role inside the cloud service management (Zhou, 2020). It is also called adversary type two and represented as A2.

Following (D. Boneh, 2004) work, (Baek, Safavi-Naini, & Susilo, 2008) have been proposed a notion of keyword guessing attack and secure channel free PEKS scheme. The notion of KGA realizes the fact that the space of the keyword used is limited in practice. These are due to fact that the people usually select the keyword that is easy to remember. The notion of secure channel-free is the removal of secure channels for trapdoors between the data receiver and cloud service provider. Furthermore, there is no complete definition that captures secure channel-free PEKS schemes that are secure against chosen keyword attacks in their works. Later on, Chao et al (Wu T. M., 2017) demonstrated that their Certificate-less Designated Server Based Public Key Encryption with Keyword Search (CL-dPEKS) schemes (Baek, Safavi-Naini, & Susilo, 2008), (Wu, Meng, Chen, Liu, & Pan, 2016) suffered from KGA on ciphertext and trapdoor by the outside adversary. Chao et al (Wu T. M., 2017) also argued research works to design a new security model and secure scheme to overcome the known attacks.

To overcome the problem of the server storing trapdoor, Baek et al. (Baek, Safavi-Naini, & Susilo, 2008) and Bosch *et al*. (Bosch, 2014) suggested that "refreshing keyword will improve the vulnerability of KGA on searchable encryption". Baek *et al*. (Baek, Safavi-Naini, & Susilo, 2008) argued that research works need to take on finding an efficient and convenient way to refresh frequently used keywords. The idea behind refreshing keyword search is to generate a trapdoor that is only valid in a specific time interval (Bosch, 2014) (Baek, Safavi-Naini, & Susilo, 2008). Considering the E-mail communication, the data sender defines the time interval for trapdoor generation by the data receiver. Every time the data receiver generates its trapdoor and search information from the cloud service provider, the trapdoor information is made to expire after the time defined by the data sender. Both Bosch *et al* (Bosch, 2014) and Baek et al (Baek, Safavi-Naini, & Susilo, 2008) suggested that searchable encryption with the refreshing keyword will improve the security of public encryption with keyword search. The literature shows that there is a gap in conducting research work in PEKS with a refreshing keyword and demonstrate its security.

Certificate-less searchable encryption with a refreshing keyword search can be defined as a public-key cryptosystem based on identity-based encryption and refreshing keyword search. The concept behind refreshing keyword search is to generate a trapdoor that is only valid in a specified time frame by the data sender. This helps to generate different trapdoors for the same keyword by attaching time information to the trapdoor indicating its validity. It also limits the time in which an adversary can launch KGA by captured trapdoors and is unable to distinguish active trapdoors from expired trapdoors. This research aims to design certificate-less searchable encryption with a refreshing keyword search CL-SERKS scheme. This needs to limit the duration of time the trapdoor remains active and to refresh keywords every time searching is conducted, by attaching time information to encrypted data and keywords.

1.1 Our contributions

In this paper, we first propose a CL-SERKS scheme and then prove its security in the random oracle model under the bilinear Diffie-Hellman problem assumption. The proposed CL-SERKS scheme refreshes the keyword by attaching time information to the encrypted keyword and the encrypted document. Similar to other certificate-less primitives, the proposed certificate-less keyword search scheme leverages the identity as the user's partial public key and eliminates the key escrow problem. We demonstrate that our scheme can resist adaptive chosen keyword attacks even in the presence of both two types of adversaries. Finally, the performance of the proposed scheme is evaluated in terms of computational and communication costs.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we present some preliminaries, which give background information on concepts that are used to design the proposed scheme. In Section 4, we present the system model of the proposed scheme. Section 5 describes the proposed scheme, security model and security proof of the proposed scheme. In Section 6, we present the performance analysis of our proposed scheme. Finally, we conclude the paper in Section 7.

2. Related work

In this section, we discuss related work conducted in the field of certificate-less searchable encryption with keyword search (PEKS).

Peng et al (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) introduced the concept of certificate-less public key encryption with keyword search, a key part of searchable encryption for both protecting data and providing operability of encrypted data. The authors' eliminated the problem of key escrow for the first time. They designed a certificate-less PEKS (CL_PEKS) scheme in an email system and constructed a secure channel-free scheme. The scheme only supports a single keyword search function. They proved its security under the bilinear Diffie–Hellman assumption. However, the scheme was later found out to be vulnerable to attacks involving a malicious key-generation-center and an offline keyword guessing attack (Wu, Meng, Chen, Liu, & Pan, 2016).

Mima et al (Ma, He, Kumar, Choo, & Chen, 2017) designed a certificate-less searchable public-key encryption with multiple keywords (SCF-MCLPEKS) scheme for IoT (Internet of Things). They demonstrated the security of the scheme in the random oracle model against both types of adversaries, where type I adversary one is given the power to select a random public key as a replacement for the user's public key and type II adversary two is capable to learn the master key. The performance of the proposed scheme was evaluated in terms of communication and computational cost. Zheng et al. (Zheng, Li, & Azgin, 2015) integrate certificate-less cryptography with a keyword search on encrypted data. They presented a concrete construction and proved its security under the decisional linear assumption in the standard model.

Wu et al. (Wu, Meng, Chen, Liu, & Pan, 2016) proved that the certificate-less searchable public-key encryption scheme of Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) cannot resist a malicious PKG attack and an offline keyword guessing attack. Therefore, the main contribution of the research work of (Wu, Meng, Chen, Liu, & Pan, 2016) was to address the vulnerability of the CL-PEKS schemes.

Even though there was a proposed certificate-less-based keyword searchable encryption, the trapdoors for a keyword were never refreshed. For this reason, their schemes are vulnerable to KGA, which allows an attacker to recover the keyword from the trapdoor (Zhou, 2020). To reduce the vulnerability of certificate-less searchable encryption with keyword search to keyword guessing attacks, we propose a CL-SERKS scheme that resists KGA.

3. Preliminaries

3.1 Bilinear pairing

Let G_1 is an additive cyclic group and G_T is a multiplicative cyclic group having the same order q with G1, we build bilinear pairing e: $G_1 \times G_1 \rightarrow G_T$ is a map. This mapping satisfies the following properties.

1. Bi-linearity: $\forall$a, b $\in Zq^*$ and $\forall$K, L $\in G1$, e (aK, bL) = e (bL, aK) = e (K, L) ^{ab}, where Z be a set of integers and a, b $\in Zq^*$ is a cyclic group of prime number.
2. Non-degenerate: there exists K, L $\in G_1$ so that e (K, L) $\neq 1 \in G_T$.
3. Computable: $\forall$K, L $\in$ G1, there is an efficient algorithm to compute e (K, L).

3.2 Bilinear Diffie-Hellman problem assumption

Let e: $G_1 \times G_1 \rightarrow G_T$ be a bilinear pairing. Suppose that we have a generator P of G_1 and some known points P, aP, bP, $cP \in G1$, where a, b, $c \in Zq*$ are unknown numbers, the bilinear Diffie-Hellman (BDH) problem is to compute $e(P, P)^{abc} \in G_T$. It can be stated that BDH problems are intractable provided that any polynomial-time algorithm has a negligible advantage in computing BDH problems

$$(\Pr[(aP, bP, cP) = e\,(P, P)^{abc}]) \leq \varepsilon. \quad (1)$$

4.System model

Next, we describe the system model of our proposed certificate-less searchable encryption with a refreshing keyword search scheme, which has four entities, namely: a cloud server, a data sender, a data receiver, and a key generation center (KGC).

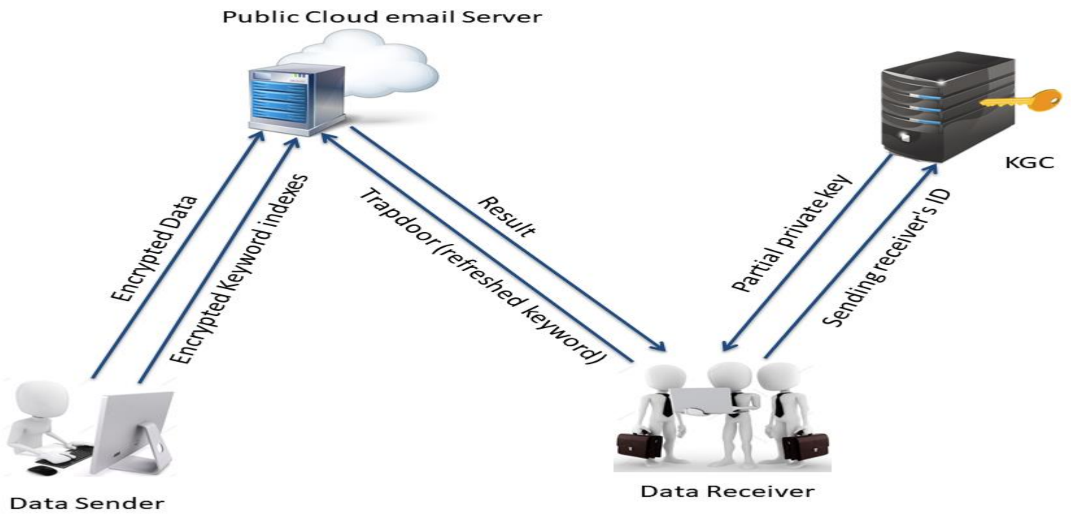

Figure 1 The CL-SERKS model

KGC is responsible for generating system keys.
Data sender uses the receiver's and server's public keys to encrypt the data and the index of keywords contained in the data. Once this has been performed, the data owner can store the encrypted data and encrypted keyword indexes in the cloud service provider.
Data receiver obtains his/her partial private key from the KGC and generates the trapdoor of keywords that he/she wants to search and sends it to the cloud server.
The cloud service provider is responsible for processing data, such as storing and searching data for a user.

5.Proposed scheme

In this section, we present the proposed CL-SERKS scheme, security model and security proof of the proposed scheme

5.1 Proposed CL-SERKS scheme

The CL-SERKS scheme consists of the following polynomial-time probabilistic algorithms:
Setup (k): suppose q is a large prime number, G is a group with order q. Let P denote a generator of G, and choose four different cryptographic hash functions $H1$, $H2$, $H3$: $\{0, 1\}^* \rightarrow G1$ and $H4$: $G_T \rightarrow \{0, 1\}^n \in Zq*$ where n is a fixed-length binary string, KGC performs the following steps

1. Select two cyclic groups G_1, G_T with the same order q and choose a bilinear pairing e: $G_1 \times G_1 \rightarrow G_T$
2. Choose a generator $P \in G_1$ and select a number $s \in Zq^*$ randomly
3. Compute $P_{pub}= sP \in G_1$. Let s be the master key,
4. Publishes public parameter $prms = (k, G_1, G_T, q, P, P_{pub}, H1, H2, H3, H4)$ and keep master key s as secret. And moreover, the refresh keyword space $w = \{0, 1\}^* \in G_1$ and refreshed encrypted keyword space is $C_w = \{0, 1\}^n \in G_T$.

Extract partial private key (D_{ID}): KGC receives data user's digital identifier $ID \in \{0, 1\}^*$ as an input and computes $Q_{ID} = H1(ID)$. Then compute data user's partial private key $D_{ID} = sQ_{ID}$
Set secret value (X_{ID}): The data sender and data receiver input their identities $ID \in \{0, 1\}^*$, then the data user chooses $X_{ID} \in Zq*$ randomly as its secret value.
Set private key (SK_{ID}): The data user input X_{ID} and $D_{ID.}$ Then compute $SK_{ID} = (X_{ID}, D_{ID})$
Set public key (PK_{ID}): Input public parameter and secret value X_{ID}, compute $PK_{ID} = X_{ID}$ P
Encryption: Let w = $\{wi \mid 1 \leq i \leq n\}$ represent a set of refreshing keywords. In our case, a keyword is refreshed by the frequently used keyword by attaching date information to the encrypted document, for example, a keyword $w = w||24/02/2021$ represents a refreshing keyword where 24/02/2021 denotes "24 February 2021" and, owner of a data sends this refreshed keyword to the public cloud server. A data sender executes as follows to encrypt refresh keyword $wi \in W$ after obtaining public parameter $prms$, data user's public key PK_{ID} and data user's or receiver's identity ID as an input:

1. Compute $Q_{ID} = H1(ID)$
2. Choose a random number $ri \in Zq*$. Then perform $Ui = riP$.
3. Compute $Ti = e(ri(H2(w||24/02/2021)), PK_{ID}) e(riQ_{ID}, P_{Pub}) e(ri(H3\ (w||24/02/2021)), P)$
4. Perform: $Vi = H4\ (T_i)$ (2)

Final encrypted document or cipher-text is the output and given by: $C = \{C1, C2\ \ldots\}$
Where $Ci = (Ui, Vi)$ (3)
Trapdoor: In this step, the algorithm receives public parameter $prms$, a refreshing keyword and, the data user's Secret Key SK_{ID} as an input. The trapdoor is performed by the data user as:
$T_w = X_{ID}\ (H2\ (w||24/02/2021)) + D_{ID}$ (4)
Test: At the test stage, the public cloud server receives public parameter $pmrs$, refreshed keyword w's trapdoor $Tw's$ and refreshed keyword encrypted document Cw as input and Performs:
$Vi = H4\ (e(T_w + (H3\ (24/02/2021)), Ui))$ (5)
If the equation (5) is correct, the output is "1"; else outputs "0". Assume that $w = wi$, wherein $i \in \{1, 2\ \ldots\ n\}$, we show that the CL-SERKS scheme fulfills the computational consistency as given below:
$H4\ (e(T_w + (H3\ (wi\ ||24/02/2021)), Ui))$

$=H4\ (e(X_{ID}(H2\ (w\|24/02/2021)) + D_{ID} + (H3\ (w\|24/02/2021)), riP)).$
$= H4\ (e(X_{ID}(H2\ (w\|24/02/2021)), riP)\ e(D_{ID}, riP)\ e(H3\ (w\|24/02/2021), riP))$
$= H4\ (e(ri(H2\ (w\|24/02/2021)), X_{ID}P)\ e(sQ_{ID}, riP)\ \ e(H3\ (w\|24/02/2021), riP))$
$=H4\ (e\ (ri\ (H2\ (wi\ \|24/02/2021)), PK_{ID})\ (riQ_{ID}, Ppub)\ (ri\ (H3\ (wi\ \|24/02/2021)), P))$
$=Vi$ (6)

5.2 Security model

In the certificate-less cryptosystem, there are two types of adversaries. Type I adversary one, is denoted as A1 and has no master key but can replace anyone's public key, and type II adversary two, is denoted as A2 and holds the master key but cannot replace anyone's public key. The security model is defined through two games played between adversaries and a challenger ch. The challenger is an honest data user in the cryptosystem which implies that it works following a predefined cryptographic framework. An adversary is in charge of communicating to the system by placing a query to the oracle, that orders a challenger to answer the queries to either of the adversary type $A1$ or adversary type $A2$ and its main goal is to break the proposed system. The CL-SERKS scheme mainly considered ciphertext indistinguishability and trapdoor indistinguishability.

5.2.1 Ciphertext indistinguishability

Ciphertext indistinguishability (IND-CKA) ensures that the ciphertext reveals no information about the underlying keyword to the cloud server. The CL-SERKS is said IND-CKA secure if the advantage of winning in the following *Game 1* and *Game 2* are negligible
Game 1: In this game, we set the semi-trusted cloud service provider as the adversary A1.
Setup: The challenger performs the setup algorithm so that it obtains public parameter $prms$ and master key s. Then it sends public parameter $prms$ to the $A1$ and keeps master key s privately.
Phase 1: $A1$ performs the oracle query as follows:
Hash query: $A1$ can query all hash algorithms and get corresponding answers.
Partial private key extract query: Given identity ID_i, Ch executes partial private key extract algorithm to acquire D_{ID} send to $A1$.
Request public key query: Given identity ID_i, Ch produces PK_{ID} and sends it to the $A1$.
Substitute public key query: $A1$ can select a random value in place of the data user's or receiver's PK.
Private key extract query: Given identity ID_i, ch calculates the corresponding private SK_{ID} to $A1$.
Trapdoor query: Given the keyword w_i with identity ID_i, ch calculates the corresponding trapdoor T_{Wi} to the $A1$.
Challenge: $A1$ chooses keyword (w_0, w_1) and identity ID_i, which is expected to challenge, ch randomly chooses a ∈ {0,1} and runs encryption algorithm to produce a target keyword cipher-text C_a to $A1$.
Phase 2: $A1$ can issue the polynomial query like phase 1, but cannot make a trapdoor query with $wi \neq (w_0,w_1)$.
Guess: Finally, the $A1$ yields $b' \in \{0, 1\}$. We say adversary $A1$ wins this game if $b'= b$.
Game 2: In this game, we set the semi-trusted KGC as the adversary A2.
*Setup***:** ch run the setup algorithm is executed to get public parameter $prms$ and master keys s of the system. Then adversary $A2$ receives public parameter $prms$ and master key s from Ch.
Phase 1: $A2$ can adaptively issue public key queries, extract private key queries, perform hash queries, and trapdoor queries.
*Challenges***:** $A2$ chooses keywords ($w_0\ w_1$) and identity ID_i to challenge, Ch selects a ∈ {0,1} uniformly and runs an encryption algorithm to produce C_a to A2.
Phase 2: $A2$ can issue the polynomial query like phase 1, but cannot make a trapdoor query with $wi \neq (w_0,w_1)$.
Guess: The $A2$ yields $b' \in \{0, 1\}$. We say adversary $A2$ win this game if $b' = b$.

5.2.2 Trapdoor Indistinguishability

Trapdoor indistinguishability guarantees that the cloud service provider cannot obtain any information about the keyword from a given trapdoor.

If A1 and $A2$ advantage of winning in the following games *Game 3* and *Game 4* are negligible respectively, we can say that the scheme is satisfied with the trapdoor indistinguishable under the adaptive chosen keyword attack.

Game 3: The interaction between A1 and the Ch is as follows:

Similar to Game 1, we set the semi-trusted cloud server as the adversary A1.

Setup: The ch outputs the system parameters by running the setup algorithm, where ch does not know the master key s.

Phase 1: $A1$ can adaptively issue query as follow:

Hash query: $A1$ can query the hash algorithm and get the corresponding answer.

Extract partial private key query: Given the identity ID_i, the ch calculates the corresponding partial private key to the $A1$.

Public-Key query: Given the identity ID_i, the ch calculates the corresponding public key to the $A1$.

Replace public key query: $A1$ can replace the public using his choice.

Encryption query: Given the keyword wi with identity IDi, ch calculates the corresponding ciphertext Ci = (Ti, Vi) to $A1$.

Challenges: $A1$ chooses keywords (w_0, w_1) and challenges identity ID_i to challenge. The ch randomly selects $b \in \{0,1\}$ and returns trapdoor search T_w to $A1$ by running the trapdoor algorithm.

Phase 2: $A1$ can issue the polynomial query like phase 1, but cannot make CL-SERKS query with $wi \neq (w0, w1)$.

Guess: $A1$ outputs $b' \in \{0, 1\}$.

Game 4: The interaction between $A2$ and ch is as follows.

Setup: The ch outputs the public parameters and the master keys by running the setup algorithm.

Phase 1: $A2$ can adaptively issue a hash query, public key query, private key query and encryption query.

Challenges: $A2$ chooses keywords (w_0, w_1) and identity ID_i to challenge. Then challenge randomly selects $b \in \{0,1\}$ and returns trapdoor search T_W by running trapdoor algorithm to $A2$.

Phase 2: $A2$ can issue the polynomial query like phase 1, but cannot make encryption query with $wi \neq (w_0, w_1)$.

Guess: $A2$ outputs $b' \in \{0, 1\}$.

5.3 Security proof

Theorem 1: The CL-SERKS scheme proposed in this research is secure semantically in random oracles against adaptive chosen keyword attacks if the problem of BDH is not breakable to resolve in probabilistic polynomial time.

Lemma 1: Supposing the existence of probabilistic polynomial-time adversary $A1$ in Game 1, which can attack the proposed CL-SERKS scheme with $\boldsymbol{\varepsilon}$ i.e. a very minimum chance of breaking the proposed scheme. We can build ch (i.e. challenger algorithm) to compute the problem BDH with the advantage ε. Suppose qH1, $qH4$, $qTrap$, $qEpart$ and $qEpriv$ represent the numbers of $H1$ query, $H4$ query, trapdoor query, partial private key extract query and private key extract query respectively. The challenger algorithm will be built to compute the problem of BDH with the advantage:

$$\varepsilon' \geq \frac{\varepsilon}{qH1qH4}\left(1-\frac{1}{qH1}\right)^{qEpart + qEpriv + qTrap} \tag{7}$$

Ch simulates challenger and replies to all the queries from adversary $A1$ in such a way that it uses the $A1$ to compute the problem.

Proof: Following Lemma 1, it is necessary to show that our framework is secure. Hence, a BDH problem difficult to solve chosen, and it is reduced to the security of our system. Ch inputs a BDH problem be the instance of (P, aP, bP, cP), the goal is to challenge A1 to compute $e(P, P)^{abc}$.

Setup: The Ch algorithm is provided with two groups G_1 and G_T of equal order q, the generator P of G_1 and a bilinear map e: $G_1 \times G_1 \rightarrow G_T$, an algorithm Ch additional select four hash functions: $H1$, $H2$, $H3$:$\{0, 1\}^* \rightarrow G_1$ and $H4$: $G_T \rightarrow \{0,1\}^n \in Zq^*$, n denotes a binary string of fixed length. For instance, challenger set $P_{pub}= aP \in G_1$ for unknown value $a \in Zq^*$ and picks $IDi \in \{0, 1\}^*$ randomly as a digital identifier challenge. Then the Ch packages the parameters as *prms* (k, G_1, G_T, q, P, P_{pub}, $H1$, $H2$, $H3$, $H4$)

and returns public parameter *prms* to the adversary $A1$ where $H1$, $H2$, $H3$, $H4$ are random oracles organized by the challenger algorithm. Then, the ch algorithm is required to provide the responses because of an $A1$ queries without understanding a master secret key s and execute the following queries in phase one.

H1 Query: A ch maintains a hash list called *H1-List* containing tuples *(IDi, δi, Q_{IDi})* that is empty initially. When the identity IDi is submitted for query, the challenger checks whether IDi is already in the hash list *H1-List*. If the algorithm challenger ch does find $ID\boldsymbol{i}$ already in the tuple (IDi, δi, Q_{IDi}) in *H1-List* then the challenger algorithm produces Q_{IDi} and sends to $A1$. If $IDi = IDI$, the challenger chooses an arbitrary number $\sigma i \in Zq^*$ and calculates $Q_{IDi} = \sigma ibP$. Otherwise, chooses an arbitrary number $\sigma i \in Zq^*$ and calculates $Q_{IDi} = \sigma iP$. Finally, the ch adds (IDi, σi, Q_{IDi}) to *H1-List* and output Q_{IDi} to $A1$.
H2 Query: A ch maintains a list *H2-List* with tuples (wi, δi, $H2$ (wi)) that are empty initially. When adversary $A1$ asks a $H2$ query for sets of keywords w_i, a ch replies as follows and tests to check whether w_i, is already reserved in the *H2-List*. If $H2$ (wi) is already in a tuple (wi, δi, $H2$ (w_i)) in *H2-List*, then the ch returns the corresponding $H2(wi)$ to adversary $A1$. Otherwise, the ch chooses an arbitrary number $\delta i \in Zq^*$ and calculates $H2$ (wi) $=\delta iP$. Later, outputs $H2$ (wi) and ch update (wi, δi, $H2$ (wi)) to *H2-List* and send $H2$ (wi) to $A1$.
H3 Query: The ch maintains a hash list *H3-List* with the tuples (wi, αi, $H3$ (wi)) that is empty initially. When $A1$ asks a $H2$ query for sets of keywords wi, a ch checks whether it is already in the *H3-List*. If this query has been asked challenger yields the record $H3$ (wi) and submits to $A1$. Otherwise, ch chooses an arbitrary number $\alpha i \in Zq^*$ and calculates or computes $H3(wi) = \alpha iP$. Finally, the ch returns $H3$ (wi) to $A1$ and adds (wi, αi, $H3$ (wi)) to the *H3-List*.
H4 Query: The ch maintains the hash list *H4-List* containing tuples (Ti, Vi) that is empty initially. When $A1$ makes hash a $H4$ query with $Ti \in G_T$, ch answers and tests if Ti is already stored in the *H4-List*. If Ti is in the tuple (Ti, Vi) then yields Vi and returns to $A1$. Otherwise, ch chooses an arbitrary number Vi = {0, 1}. Finally, the challenger (Ti, Vi) is added to the *H4-List* and sends Vi to $A1$.
Partial private key extract query: The ch maintains a list of partial private keys referred to as $PPK\text{-}List$ containing tuples (IDi, Q_{IDi}, $_i$). When $A1$ queries for extraction of the private partial key query of IDi from $A1$, the Ch performs or replies as follows: A partial private key list, returns to D_{IDi} as a response to $A1$, if there exists is a tuple containing the form of (IDi, Q_{IDi}, $_{IDi}$). Otherwise, if $IDi \neq IDI$ query the $H1$ hash $List$ for a tuple form (IDi, σi, D_{IDi}). The ch calculates $D_{IDi} = \sigma iP_{Pub} = \sigma iaP$, adds ($IDi$, Q_{IDi}, D_{IDi}) into the private partial key list ($PPKList$) and returns to the back D_{IDi} as a response to $A1$. If $IDi = IDI$, challenger aborts.
Request public key query: The ch maintains a public key list $PK\text{-}List$ containing tuples (IDi, Xi, PK_{IDi}). When $A1$ asks for the public key query of identity IDi, a ch answers as follows: If public key identity PK_{IDi} exits in the tuple form (ID , xi , PK_{IDi}) in the public key list, the ch sends PK_{IDi} as a response to $A1$. Else, select a random number $xi \in Zq^*$, calculate $PK_{IDi} = XiP$, adds (IDi, x_i, PK_{IDi}) into the $PK\text{-}List$ and returns $PK_{\boldsymbol{IDi}}$ to $A1$.
Replace public key query: $A1$ can substitute a data user's public keys with new random values.
Extract private key query: In this phase, it takes identity IDi as an input and the ch maintains tuples of a private key list form (IDi, xi, D_{IDi}) that exists in private key extraction query list S$K\text{-}List$. After getting a private key extraction query (IDi), ch answers as follows:
If a tuple exists (IDi, xi, D_{IDi}) on the private key list, returns to back (xi, D_{IDi}) to $A1$.
Else, $IDi \neq IDI$ executes public key request algorithm S to obtain a tuple (IDi, xi, PK_{IDi}) and also run partial private key extraction algorithm to obtain a tuple(ID_i, Q_{IDi}, D_{IDi}), modify to the private key list (IDi, xi, D_{IDi}) and returns (xi, D_{IDi}) as a response to $A1$.
If $IDi = IDI$, the ch aborts.
Trapdoor query: When $A1$ requests trapdoor query on refresh sets of keywords wi for an identity IDi, the Ch replies as follows: If $IDi = IDI$, the ch aborts. Else, recovers (IDi, xi, PK_{IDi}) from public key list $PK\text{-}List$, retrieves (IDi, Q_{IDi}, D_{IDi}) from *PPK-List* and recovers refreshed keyword (wi , δi , $H2$ (wi)) from

hash list $H2$ $List$ and calculates $Tw_i = X_{ID}H2\ (wi) + D_{ID\mathrm{i}}$. Finally, the ch returns the trapdoor Tw_i of the keyword w_i to $A1$.
Challenge: If $A1$ issues a challenge that phase one is over, select $ID*$, ∈ {0,1} with public key $PK_{\boldsymbol{ID}^*}$ and two different keyword sets $w0$, $w1$, where $w0 \neq w1$, $|w0| = |w1|$ and not requested by $A1$ in phase one, then ch executes as below: If $IDi^* \neq IDI$ then Ch aborts. If $IDi = IDI$, ch selects a ∈ {0, 1} randomly, and picks two arbitrary numbers that include $\boldsymbol{r} \in Zq^*$ and V= {0, 1}n.
Finally ch submits $C^* = (rcP, V)$ to adversary $A1$. If $C_{\mathrm{a}} = (rcP, V)$ is a meaningful cipher-text, then it is calculated as follows:
$V = H4\ ((\delta P, xiP)^{rc}\ (\sigma ibP, aP)^{rc}\ (aP, P)^{rc})$
$= H4\ ((P, P)^{\delta xi\, rc}\ (P, P)^{\sigma i a^{b} rc}\ (P, P)^{arc})$
$= H4\ ((P, P)^{rc\,(\delta xi+\sigma)}\ (P, P)^{\sigma iab\, rc})$ (8)
More queries: $A1$ can execute or perform additional trapdoor queries on keyword sets wi where $w_0 \neq w_1$ and $wi \neq w1$, challenge responds as above. Additionally, an algorithm ch answers to $A1$ in a similar way as phase one under the restrictions defined below:

$A1$ is not able to query private key queries set of keywords on the challenge identity IDi^*.

$A1$ is not able to place a partial private key extraction query on the challenge identity $IDi*$ if the public key of the challenge identity IDi^* was substituted before the challenge phase.

$A1$ is not able to put a trapdoor query on (Tw_0, IDi^*) or trapdoor query on (Tw_1, IDi^*) without the challenge phase.

Guess: Lastly, $A1$ yields $b' \in \{0, 1\}$ as its guess. In this case, ch can select a pair (Tw, Vi) arbitrarily from the hash list $H4$ $List$ and $e(P, P)^{abc}$ can be calculated as follows:
Lastly, $A1$ yields $b' \in \{0, 1\}$ as its guess. In this case, ch can select a pair (T_i, V_i) arbitrarily from the H4-List and $e\ (P, P)$ can be calculated as follows:
$e\ (P, P)^{abc} = (\ T\ /e(P,P)^{rc(\delta xi^{+}\alpha))} \frac{1}{\mathrm{r\sigma i}}$
$= (\ e(P,P)^{rc(\delta xi^{+}\alpha)}\ e(\ P,P)^{\alpha brc\sigma i} /e(P,P)^{rc\delta xi^{+}\sigma i}\ )\ ) \frac{1}{\mathrm{r\sigma i}}$
$= e\ (P, P)^{abc}$ (9)

6.Performance evaluation
We provide performance evaluations of our proposed certificate-less searchable encryption with a refreshing keyword search system with some existing research works such as Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) and Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017) in terms of computational time and security property.

6.1 Computation cost
The notations and the executing times used in the evaluation are defined in Table 1. The evaluation was performed on a personal computer (Dell with an i5-4460S 2.90GHz processor,4G bytes memory and Windows 8 operating system) using the MIRACL library (ltd., 2016).

Table 1 *Notations and execution times (ms)*

Notations	Description	Times(ms)
T_{sm}	a scalar multiplication execution time	2.165
T_{bp}	a bilinear pairing execution time	5.427
T_H	a Hash-to-point execution time	5.493
T_h	a general hash function execution time	0.007
T_{pa}	a point addition execution time	0.013

Table 2 shows the computational cost for Peng et al's. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) , Mima et al's. (Ma, He, Kumar, Choo, & Chen, 2017) scheme, and our proposed scheme.

Table 2 *Comparison of computational costs*

Comparison criteria	Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014)	Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017)	Our proposed scheme
keyGen	$2T_H + 8T_{sm}$ =28.306	$2T_H + 4T_{sm}$ =19.646	T_H+3T_{sm}=11.988
Encryption	$3T_H + 2T_h + 5T_{sm}+ 3T_{bp}$=43.599	$3T_H + T_h + 4T_{sm}+ 3T_{bp} + T_{pa}$=41.433	$2T_H+T_h+4T_{sm}+3\ T_{pb}$ =35.927
Trapdoor	$T_H + T_h + 3T_{sm}$ =11.995	$T_H + T_{sm} + T_{pa}$ =7.671	$T_H+T_{sm}+T_{pa}$=7.7
Test	$T_h + T_{sm} + 2T_{pa}+ T_{bp}$=7.625	$2_{TH} + T_h + T_{sm}+2T_{pa} + T_{bp}$=18.611	$T_H+\ +T_h+T_{sm}\ +T_{pa} + T_{pb}$=13.105

The computational cost of our proposed CL-SERKS scheme is lower than the Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) and Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017) scheme's as shown in table 2 and figure 2 except in test phase execution time is higher than Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) scheme despite our proposed scheme provides refreshing keyword search.

6.2 Communication cost

We let |G1| denote the bit-size of a point in Group G

$|\mathbb{Z}q|$ denote the bit-size of a number in $\mathbb{Z}_q$

|PK| denote the bit-size of PK

|C| denote the bit-size of cipher-text

|T| denote the bit size of the trapdoor respectively. Table 3 shows the comparison of communication cost schemes in Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014), Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017) and the proposed scheme.

Table 3 *Comparison of communication cost*

Comparison criteria	Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014)	Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017)	Our proposed scheme

Size of PK	4|G1|	2|G1|	|G1|
Size of CT	|G1| + |$\mathbb{Z}q$|	|G1 | + |$\mathbb{Z}q$|	|G1 | + |$\mathbb{Z}q$|
Size of TD	3|G1|	|G1|	|G1|

It is also observed that the overall communication cost of the proposed system is less than both schemes in Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014) and Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017) .

6.3 Security property

Table 4 *Comparison of security properties*

Comparison criteria	Peng et al. (Yanguo, Jiangtao, Changgen, & Zuobin, 2014)scheme	Mima et al. (Ma, He, Kumar, Choo, & Chen, 2017) scheme	Our proposed scheme
Certificate management problem	Yes	Yes	Yes
Key escrow problem	Yes	Yes	Yes
Refreshing keyword	No	No	Yes
Trapdoor security	No	No	Yes

Yes: denotes that the system meets the security requirements.
No: denotes that the system does not meets the security requirements

7. Conclusion

In this research, we proposed the CL-SERKS scheme. The proposed CL-SERKS scheme overcomes the issue of the server storing trapdoor for keywords in the system when the user performs a keyword search by his trapdoor on the cloud server that occurs in existing PEKS. This is achieved by attaching time date information to the encrypted data and keyword. It primarily improves the security of the outsourced data by reducing the chance for the cloud to guess the full keyword information that enables to access the ciphertext. We also prove that the designed scheme is secure against adaptive chosen keyword attacks in the random oracle model under bilinear Diffie-Hellman (BDH) problem assumption. The performance of the proposed CL-SERKS method is also provided and the results are compared with some of the existing schemes in terms of the computational and communication cost. An interesting open problem is to design CL-SERKS considering advanced search functions such as conjunctive, disjunctive, boolean and fuzzy keywords search over a real-world e-mail dataset.

References

Al-Riyami. (2003). Certificateless public key cryptography. *In Proceedings of the 2003 International Conference on the Theory and Application of Cryptology and Information Security.* Berlin/Heidelber Germany.

Baek, J., Safavi-Naini, R., & Susilo, W. (2008). Public key encryption with keyword search revisited. *In Proceedings of the International conference on Computational Science and Its Applications.* Perugia, Italy.

Bosch, C. H. (2014). A survey of provably secure searchable encryption. *ACM Computing Surveys (CSUR)*, 26.

D. Boneh, G. D. (2004). Public Key Encryption with Keyword Search. *Advances in Cryptology (EUROCRYPT 2004).*

J. Li, X. L. (2017). KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage. *IEEE Transactions on Services Computing*.

Kamara S, P. C. (2012). Dynamic searchable symmetric encryption[C]. *Proceedings of the 2012 ACM conference on Compute rand communications security*.

Kamara, S., & Lauter, K. (2010). Cryptographic cloud storage. *In Proceedings of the 2010 International Conference on Financial Cryptography and Data Security.* Tenerife, Canary Islands, Spain,.

ltd., S. s. (2016). *miracl library," http://www.shamus.ie/.* Shamus software ltd.

Ma, M., He, D., Kumar, N., Choo, K., & Chen, J. (2017). Certificateless searchable public key encryption scheme for industrial internet of things. *IEEE Trans. Ind. Inform*.

Song D X, W. D. (2000). Practical techniques for searches on encrypted data. *Proceedings on Security and Privacy*.

Wang, G. L. (2017). *IDCrypt: A multi-user searchable symmetric encryption scheme for cloud applications*.

Wu, T. M. (2017). Comments on Islam et al.'s certificateless designated server based public key encryption with keyword search scheme. *In International Conference on Genetic and Evolutionary Computing.* Springer, Singapore.

Wu, T., Meng, F., Chen, C., Liu, S., & Pan, J. (2016). On the security of a certificateless searchable public key. *International Conference on Genetic and Evolutionary Computing.* Berlin/Heidelberg, Germany.

Yanguo, P., Jiangtao, C., Changgen, P., & Zuobin, Y. (2014). Certificateless public key encryption with keyword. *China Commun*.

Zheng, Q., Li, X., & Azgin, A. (2015). Certificateless keyword search on encrypted data. *In Proceedings of the International Conference on Network and System Security.* Berlin/Heidelberg, Germany.

Zhou, Y. L. (2020). Public Key Encryption with Keyword Search in Cloud. *A Survey. Entropy*.