Certificateless Public Key Encryption with Keyword Search

PENG Yanguo[1], CUI Jiangtao[1], PENG Changgen[2*], YING Zuobin[1]

[1] School of Computer Science and Technology, Xidian University, Xi'an 710071, Shaanxi Province, P. R. China
[2] College of Science, Guizhou University, Guiyang 550025, Guizhou Province, P. R. China

Abstract: *Public Key Encryption with Keyword Search* (PEKS), an indispensable part of searchable encryption, is stock-in-trade for both protecting data and providing operability of encrypted data. So far most of PEKS schemes have been established on *Identity-Based Cryptography* (IBC) with key escrow problem inherently. Such problem severely restricts the promotion of IBC-based *Public Key Infrastructure* including PEKS component. Hence, C*ertificateless Public Key Cryptography* (CLPKC) is efficient to remove such problem. CLPKC is introduced into PEKS, and a general model of *Certificateless PEKS* (CLPEKS) is formalized. In addition, a practical CLPEKS scheme is constructed with security and efficiency analyses. The proposal is secure channel free, and semantically secure against adaptive chosen keyword attack and keyword guessing attack. To illustrate the superiority, massive experiments are conducted on Enron Email dataset which is famous in information retrieval field. Compared with existed constructions, CLPEKS improves the efficiency in theory and removes the key escrow problem.

Keywords: provable secure; certificateless public key encryption; keyword search

I. INTRODUCTION

Data as a Service (DaaS), as a main function of cloud computing, provides an assurance that data is provided on demand to user regardless of geographic or organizational separation of provider and consumer. To reduce costs and promote efficiency, organizations have been focusing on outsourcing their storage and computing needs currently. In DaaS, *Public Key Encryption with Keyword Search* (PEKS), as a fundamental searchable encryption component in *Public Key Infrastructure* (PKI), is efficient to both safeguard outsourced data (through encryption) and provide operability over encrypted data. Hence, PEKS has been introduced to eliminate secure concern in DaaS environment. The most urgent difficulty today in developing secure cloud computing is not promoting the efficiency of a kind of secure algorithm. Rather, it is the enhancement and optimization of infrastructure to support widespread and practical functions.

In existing Internet and cloud environment with unlimited resource, it is intractable to manage certificates in PKI. Moreover, in resource-constrained environment (such as internet of Things, mobile network, et al), it becomes an impossible mission. *Identity-Based Cryptography* (IBC) [1] provides an approach to derive participant's public key from his unique identifier, such as cellphone number, E-mail address, et al. It reduces the cost of certificate management through binding participant's identity and public key. Be-

CLPKC is introduced into PEKS, and a general model of Certificateless PEKS (CLPEKS) is formalized. In addition, a practical CLPEKS scheme is constructed with security and efficiency analyses.

cause *Private Key Generator* (PKG) holds all the participants' key pairs inherently, it leads to key escrow problem. Once PKG is compromised, sensitive information of users will be revealed completely along with the private key. It severely restricts the promotion of PKI including PEKS component. Furthermore, most of proposed PEKS schemes so far were established on IBC with key escrow problem inherently [2-5].

Certificateless Public Key Cryptography (CLPKC) [6] is well-known for the ability of removing key escrow problem fundamentally. In CLPKC settings, user's private key is cryptographic computed with its secret value generated by itself and a partial private key from *Key Generation Center* (KGC). Naturally, KGC does not have access to user's private key. In CLPEKS other than traditional PKI and IBC-PKI, user's private key can be determined after its public key has been generated and used. It brings out the certificateless characteristic much more practical for DaaS. Hence, certificateless public key encryption and signature schemes have been developed[7,18], and applied to DaaS environment [8-12]. So far, no construction is proposed to support both PEKS and certificateless characteristic in DaaS.

Scenario. Cloud sharing and E-mail system are two most typical applications of DaaS in which search ability is indispensable. In the following contents, E-mail system is as the application to illustrate our presentation. There are two physical entities in E-mail system, *client* and *server*. Further *client* is divided into two logical entities, *sender* and *receiver*. Note that *sender* and *receiver* are seldom online simultaneously.

Sender sends mails to Receiver, and retrieves mails from *server*. *Server* stores and manages the mails for all *clients*. When requesting mails for *receiver*, he sends a keyword "unread" to *server*. *Server* will transmit the mails of "unread" *receiver* wants. Mail is the message to delivery through an insecure channel and is characterized as a multi-tuple *<content, sender, receiver, title, subject, time, ...>*. In the tuple, some values are viewed as keywords, such as *name*, *subject*, *title*, *key point*, et al.

Naturally, *receiver* retrieves mails through various devices, such as laptop, cellphone, portable equipment, et al. As before, some are resource-constrained. Mails also contain massive number of private information. Necessarily, public key encryption is introduced into E-mail system to provide data security. However, previous works on such aspect do not consider certificate management and resource-constrained environment. PEKS in CLPKC setting is a considerable way suitable for such scenario.

Our Contribution. We formalize and design a mechanism called *Certificateless Public Key Encryption with Keyword Search* (CLPEKS) which removes key escrow problem in previous PEKS works and possesses certificateless characteristic. Such mechanism reduces the complexity of certificate management, makes progress of CLPKC and is more applicable in DaaS environment. We mainly do the following contributions:

- A CLPEKS model and its security model are provided to guarantee the functionality and security of proposed CLPEKS scheme;
- A practical CLPEKS scheme, which is secure channel free, is proposed and is proved semantically secure against adaptive chosen keyword attack and keyword guessing attack;
- We illustrate the superiority of proposed CLPEKS scheme by massive experiments conducted on a real-life data set.

In this paper, we start from preliminaries and assumptions in section II and describe models used through the paper in section III. In section IV, the proposed CLPEKS construction is shown in detail, and then the correctness, security and efficiency analyses are stated in section V. Our ultimate goal is to design a novel practical CLPEKS scheme and exhaustive experiments are shown in section VI to demonstrate the superiority of CLPEKS.

Finally, this paper concludes with a discussion of current work and future research consideration in section VII.

Related Work

PEKS, an indispensable part of searchable encryption, makes encrypted data operable inherently and is widely used in cloud computing. There are considerable works devoted to PEKS for promoting efficiency.

Searchable encryption was proposed by Song. et al [13] based on stream cipher and symmetric encryption. The solution requires very little communication between the user and the storage provider, and only one round of interaction. But it is impossible to share an encryption key before transmit mails between clients in insecure channel, so it is not suitable to the scenario aforementioned. PEKS was proposed by Boneh. et al [14] based on bilinear pairing. In PEKS, a public/private key pair <*SK,PK*> is distributed to each *client* before sending and receiving mails. When sending a mail to *receiver*, *sender* encrypts the mails and associated keywords with *receiver's* public key *PK*. *Receiver* retrieves mails from *server* with keyword *w*'s trapdoor T_w. It is proved secure against adaptive chosen keyword attack.

Nevertheless Baek. et al points out that in [14] the trapdoor must be transmitted in secure channel. In [2], they proposed two schemes removing secure channel and providing keywords fresh ability. Rhee. et al first proposed a *Secure Channel Free Public Key Encryption with Keyword Search* (SCF-PEKS) scheme supporting a designated tester, and secondly proposed a general construction for PEKS scheme supporting a designated tester in [3] and [4] respectively. Xu. et al [5] proposed a *Public Key Encryption with Fuzzy Keyword Search* (PEFKS) that is secure under *Keyword Guessing Attack* (KGA) [15]. However most of these schemes have drawback of key escrow problem and certificate management.

II. Preliminaries and Assumptions

In this section, we first review some preliminaries used in this paper. Then, bilinear pairing, as a basic tool in our construction, is described.

2.1 Preliminaries

Suppose that G_1 is an additive cyclic group of prime order q. P is a generator in G_1. $\hat{e}$ is a computable bilinear map $\hat{e} : G_1 \times G_1 \rightarrow G_2$ as described below. With the setup above, the BDH and GBDH problem are defined:

Bilinear Diffie-Hellman(BDH) Problem: Given $P \in_R G_1$, $\hat{e}$, as well as aP, bP and cP for unknown $a, b, c \in_R Z_q$, compute $\hat{e}(P, P)^{abc}$.

Generalized BDH (GBDH) problem: Given $P \in_R G_1$, as well as aP, bP and cP for known $a, b, c \in_R Z_q$, output a computable bilinear map $\hat{e}$.

2.2 Bilinear pairing

Bilinear pairing was first introduced to cryptography by Boneh. et al [16] for its high performance in security and short signature length.

Suppose G_1 is an additive cyclic group and G_2 is a multiplicative cyclic group. They have the same order q. We construct a map as following:

$$\hat{e} : G_1 \times G_1 \rightarrow G_2$$

Where G_1 and G_2 must be chosen so that there is no known algorithm for efficiently solving the BDH problem for $\hat{e}$. The map $\hat{e}$ is called a bilinear pairing if the following properties are satisfied:

Bilinear: $\forall P, Q \in G_1$ and $\forall a, b \in Z_q^*$, $\hat{e}(aP, bQ) = \hat{e}(bP, aQ) = \hat{e}(P, Q)^{ab}$;

Non-degenerate: $\exists P, Q \in G_1$, $\hat{e}(P, Q) \neq 1 \in G_2$;

Computable: $\forall P, Q \in G_1$, there is an efficient algorithm to compute $\hat{e}(P, Q)$.

Note that $\hat{e}$ is also symmetric since it is bilinear and G_1 is a cyclic group.

PG is a GBDH generator which takes a set of secure parameters k, and generates G_1, G_2 and a bilinear pairing $\hat{e}$ constructed based on G_1 and G_2. BDH problems both in G_1 and G_2 are intractable.

III. Definitions

In this section, the CLPEKS model and security model are provided.

3.1 The model

In Email system, we denote the content of mail @ as M(same as normal plaintext), and the keywords as $w_1,w_2,...,w_m$. The mail can be formalized as following:

$@=<M, w_1,w_2,...,w_m>$.

To send a mail @ in PEKS, sender sends:

$C=<C_M, C(@)>$.

where $C(@) =< CP_{w_i} >, i \in \{1,2,\ldots,m\}$.

In the generation process of $C(@)$, *sender* only use *receiver's* public key PK_A and *server's* public key PK_S. When *receiver* retrieves mails containing a certain keyword w', he generates a trapdoor $T_{w'}$ and sends it to *server*. Given CP_{w_i} and $T_{w'}$, *server* checks whether $w_i=w_j$ or not. If $w_i \neq w_j$, *server* learns nothing about w_j and @. If $w_i=w_j$, *server* sends encrypted mail C to *receiver* and still learns nothing.

In order to meet the demands in the scenario above, a non-interactive CLPEKS scheme must satisfy the following properties:

1. When producing $C(@)$, *receiver's* private key SK_R cannot be used, and *sender* should produce it without any interaction with *receiver*.

2. With holding encrypted keywords only, no statistical information will be revealed.

3. Whenever *receiver* retrieves mails containing a keyword w' using $T_{w'}$, *server* learns nothing about keywords of checked mails and w'.

4. Both *server* and *client's* private keys must be composed of partial private key and secret value of their own choice.

For ease of description, we provide the formal model for CLPEKS scheme.

Definition 1(CLPEKS). A non-interactive certificateless public key encryption with keyword search consists of 8 polynomial-time probabilistic algorithms:

1. **Setup:** Take a set of secure parameters k, and generate public parameter *prm* and *master-key*.

2. **ePPK(Extract Partial Private Key):** Take *prm*, *client u*'s identifier ID_u and the *master-key*, and construct partial private key D_u.

3. **sSV(Set Secret Value):** Take *prm* and *u*'s identifier ID_u, and output *u*'s secret value s_u.

4. **gPriK(Generate Private Key):** Take *prm*, *u*'s partial private key D_u and secret value s_u as input and generate *u*'s private key SK_u.

5. **gPubK(Generate Public Key):** Take *prm* and *u*'s secret value s_u, and output public key PK_u.

6. **CLPEKS:** Take *prm*, PK_R, PK_S and a mail @, produce an encrypted mail $C(@)$.

7. **Trapdoor:** Take *prm*, SK_R, SK_S and a keyword w, and produce a its trapdoor T_w.

8. **Test:** Take *prm*, SK_S, an encrypted $C(@)$, and a trapdoor T_w. Output 'yes' if w is included in $C(@)$ and 'no' otherwise.

3.2 Security Model

Now, we define the security for CLPEKS in the sense of semantic-security under adaptive chosen keyword attack. The properties listed in section 3.1 must be satisfied.

Suppose that an active adversary can obtain trapdoors T_{w_i} for any keyword w_i. In this strengthened assumption, the attacker should not be able to distinguish w_0 from w_1 where $w_0 \neq w_1$ and the attacker does not obtain trapdoor T_{w_0} or T_{w_1}. This idea is proposed by Boneh. et al [14].

In certificatelss cryptography [6], there exist two kinds of adversaries, **Type I** adversary $\mathcal{A}_I$ and **Type II** adversary $\mathcal{A}_{II}$. $\mathcal{A}_I$ have no access to *master-key*. However, he has access to partial private key extraction (**PPKExt**), private key generation (**PriKGen**), public key request (**PubKReq**), public key replacement (**PKRep**) and trapdoor request (**TrapReq**) actions. That means $\mathcal{C}$ honestly answers $\mathcal{A}_I$'s request of above actions. $\mathcal{A}_{II}$ does have access to master-key, and all aforementioned actions for $\mathcal{A}_I$ except **PKRep**. Both adversaries compromise *server*.

Formally, the secure against such an attacker $\mathcal{A}$ is defined using the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$.

CLPEKS security game:

Setup: Given a set of secure parameters k,

system runs **Setup** algorithm to obtain *prm* and *master-key*. The challenger $\mathcal{C}$ sends *prm* to adversary. Also, *master-key* is given to $\mathcal{A}_{II}$ only.

Request: Adversary issues a sequence of requests, including **PPKExt**, **PriKGen**, **PubKReq**, **TrapReq** and **PKRep**. These actions are asked adaptively.

Challenge: $\mathcal{A}$ sends two keywords w_0 and w_1 on which it wishes to be challenged. The challenger $\mathcal{C}$ picks up a $b \in_R \{0,1\}$, runs the **Trapdoor** algorithm to generate CP_{w_b} as response.

Attack: Finally, $\mathcal{A}$ outputs $b' \in \{0,1\}$ and wins the game if $b=b'$.

Through the whole process of game, there are several restrictions: The adversary should not run **Request** for T_{w_0} or T_{w_1} without the **Challenge** phase; **Request** can be run before or after the occurrence of **Challenge**.

In a word, if any kind of $\mathcal{A}$ wins the game, we said adversary $\mathcal{A}$ breaks CLPEKS. We define the advantage of $\mathcal{A}$ in breaking CLPEKS as

$$Adv_{\mathcal{A}(k)} = Pr[b = b'] - \frac{1}{2}.$$

Here, we give the description of **PKRep** action. Other actions, which may occur in **Request** phase, are identical to that in Definition 1 and omitted here.

PKRep: Take *prm*, $\mathcal{A}$ can replace the public key PK_u for any *client u* with any value $PK_{u'}$.

Definition 2(IND-CKA Secure)**.** In polynomial time, if there is no adversary $\mathcal{A}_I$ or $\mathcal{A}_{II}$ breaks a CLPEKS with a non-negligible function $Adv_{\mathcal{A}(k)}$, the CLPEKS is said to be secure against an adaptive chosen keyword attack.

IV. Our Construction

We build a non-interactive CLPEKS scheme from a bilinear map. The scheme works as follows.

Setup: This algorithm runs the following steps to initialize the scheme:

1. Run PG, generate G_1, G_2 of prime order $q \geq 2^k$, and a bilinear pairing $\hat{e} : G_1 \times G_1 \rightarrow G_2$.

2. Pick up a generator $P \in G_1^*$ as part of the public parameter, and a $\gamma \in_R Z_q^*$ as the *master-key*.

3. Construct 4 hash functions: H_1: $\{0,1\}^* \rightarrow G_1^*$, H_2: $\{0,1\}^* \rightarrow Z_q^*$, H_3: $\{0,1\}^* \rightarrow G_1^*$ and H_4: $G_2 \rightarrow \{0,1\}^f$.

The KGC's public key is $PK_C = \gamma P \in G_1$ and the *master-key* is $SK_C = \gamma$. The public parameter is $prm = (G_1, G_2, \hat{e}, P, PK_C, H_1, H_2, H_3, H_4)$.

ePPK: This algorithm takes as input an identifier $ID_u \in \{0,1\}^*$ from an authorized *client u* and carries out the following steps to generate partial private key:

1. Compute $Q_u = H_1(ID_u) \in G_1^*$.

2. The partial private key is $D_u = \gamma Q_u \in G_1^*$.

sSV: This algorithm picks up a $s_u \in_R Z_q^*$ as the secret value for *client u*.

gPriK: This algorithm takes as input *prm* and *u*'s secret value s_u, and generates *u*'s private key is $SK_u=(SK_{u1}, SK_{u2})$ where $SK_{u1}=s_u$ and $SK_{u1}=s_uD_u$.

gPubK: This algorithm takes as input *prm* and a secret value s_u of *client u*. *u*'s public key is $PK_u=(PK_{u1}, PK_{u2})$ where $SK_{u1}=s_uP$ and $SK_{u1}=s_uPK_c$.

According to **gPriK** algorithm and **gPubK** algorithm, both public/private key pair $<PK_R, SK_R>$ and $<PK_S, SK_S>$ of *receiver R* and *server S* are constructed. Let α be *S's* secret value. $SK_S=(SK_{S1}, SK_{S2})$ where $SK_{S1} = \alpha$ and $SK_{S2} = \alpha D_S$, and $PK_S=(PK_{S1}, PK_{S2})$ where $PK_{S1} = \alpha P$ and $PK_{S2} = \alpha PK_C$. Let β be *R's* secret value. $SK_R=(SK_{R1}, SK_{R2})$ where $SK_{R1} = \beta$ and $SK_{R2} = \beta D_R$, and $PK_R(PK_{R1}, PK_{R2})$ where $SK_{R1} = \beta P$ and $PK_{R2} = \beta PK_C$.

Note that, in our construction, s_u and PK_u can be generated and used before SK_u is determined. It brings out the certificateless characteristic. There is no bind between ID_u and PK_u, and it removes certificate management inherently.

CLPEKS: This algorithm takes as input *receiver's* public key PK_R, *server's* public key PK_S and their identity information $ID_R, ID_S \in \{0,1\}^*$. This algorithm does the following steps to encrypt the keywords w_i in mail @ where $1 \leq i \leq m$:

1. Check that both $\hat{e}(PK_{S1}, PK_C) = \hat{e}(PK_{S2}, P)$ and $\hat{e}(PK_{R1}, PK_C) = \hat{e}(PK_{R2}, P)$ hold. If not, return $\perp$ and abort.

2. For each keyword w_i where $1 \le i \le m$:

a) Pick up $r_i \in_R Z_q^*$.

b) Compute $Q_R = H_1(ID_R)$ and $Q_S = H_1(ID_S)$.

c) Compute $t_i = \hat{e}(r_i H_2(w) Q_R, PK_{R2}) \hat{e}(r_i Q_S, PK_{S2}) \hat{e}(r_i H_3(w), P)$.

d) $CP_{w_i} = [U_i, V_i]$ for $U_i = r_i P$ and $V_i = H_4(t_i)$.

3. The final cipher text for keywords is $C(@) = < CP_{w_1}, CP_{w_2}, \ldots, CP_{w_m} >$, where $CP_{w_i} = [U_i, V_i]$.

Note that we focus only on encrypting (decrypting) keyword but the mail itself. How to generate encrypted mail is beyond the matter in this paper. During the generation of CP_{w_i} for w_i, a random number r_i was introduced, so with holding encrypted keywords only, no statistical information is revealed. The second property listed in section 3.1 is satisfied.

Trapdoor: This algorithm takes as input *server's* PK_S, *receiver's* SK_R and a keyword w. The trapdoor is constructed as follows:

1. Compute $T_1 = \lambda P$.

2. Compute $T_2 = H_2(w) SK_{R2} \oplus \lambda PK_{S1}$.

3. Compute $T_3 = H_3(w) \oplus \lambda PK_{S1}$.

4. Let the trapdoor be $T_w = [T_1, T_2, T_3]$.

Test: This algorithm takes as input *server's* private key SK_S, keyword w's trapdoor T_w, and an encrypted mail $C(@)$. Verify whether T_w is included in $C(@)$ or not as follows:

1. Compute $T_2' = T_2 \oplus (SK_{S1} T_1)$.

2. Compute $T_3' = T_3 \oplus (SK_{S1} T_1)$.

3. Check that $H_4(\hat{e}(T_2' + SK_{S2} + T_3', U_i)) = V_i$ holds. If the equivalent holds, keyword w is included in $C(@)$ and output 'yes'. Otherwise, output 'no'.

Notice that, in the process of CLPEKS, *sender* produces $C(@)$ by himself using *receiver's* public key PK_R and *server's* public key PK_S. It is a basic requirement for CLPEKS, because when *sender* produces $C(@)$, *receiver* is off-line always. Hence, the first property listed in section 3.1 is satisfied.

V. Analysis

In this section, we give several theorems about correctness and security of our construction. Next, compared with other related constructions, the efficiency analysis is given.

5.1 Correctness

Theorem 1 (Correctness). Given an encrypted mail $C(@)$ belonging to *receiver* and a trapdoor T_w from himself, and $w \in @$, the equation $H_4(\hat{e}(T_w, w_i)) = V_i$ is satisfied for a specified i where $w = w_i$. Otherwise, $H_3(\hat{e}(T_w, w_i)) \neq V_i$.

Proof of theorem 1.

Let $T_1' = SK_{S2} T_1$ and we get:

$$\begin{aligned}
&H_4(\hat{e}(T_2' + SK_{S2} + T_3', U)) \\
&= H_4(\hat{e}((T_2 \oplus T_1') + SK_{S2} + (T_3 \oplus T_1'), r_i P)) \\
&= H_4(\hat{e}((H_2(w) SK_{R2} + SK_{S2} + H_3(w), r_i P))) \\
&= H_4(\hat{e}(H_2(w) \beta D_R + \alpha D_S + H_3(w), r_i P)) \\
&V_i \\
&= H_4(t_i) \\
&= H_4(\hat{e}(r_i H_2(w) Q_R, PK_{R2}) \hat{e}(r_i Q_S, PK_{S2}) \hat{e}(r_i H_3(w), P)) \\
&= H_4(\hat{e}(r_i H_2(w) Q_R, \beta PK_C) \hat{e}(r_i Q_S, \alpha PK_C) \hat{e}(r_i H_3(w), P)) \\
&= H_4(\hat{e}(H_2(w) \beta\gamma Q_R, r_i P) \hat{e}(\alpha D_S, r_i P) \hat{e}(H_3(w), r_i P)) \\
&= H_4(\hat{e}(H_2(w) \beta D_R + \alpha D_S + H_3(w), r_i P))
\end{aligned}$$

Because $w = w_i$, $H_4(\hat{e}(T_2' + SK_{S2} + T_3', U)) = V$ holds.

There are several other situations as follows:

- If there is no specified i such that $w = w_i$, then $w \neq w_i$, $H_2(w) \neq H_2(w_i)$ and $H_3(w) \neq H_3(w_i)$. So, $H_3(\hat{e}(T_2' + SK_{S2} + T_3', U)) \neq V$.
- If T_w is generated by Eve whose $SK_{E1} = \delta$, that means $V_i = H_4(\hat{e}(H_2(w) \beta D_E + \alpha D_S + H_3(w), r_i P))$. We can get

$$\begin{aligned}
&H_4(\hat{e}(T_2' + SK_{S2} + T_3', U)) \\
&= H_4(\hat{e}(H_2(w) \beta D_R + \alpha D_S + H_3(w), r_i P)) \\
&\neq H_4(\hat{e}(H_2(w) \beta D_E + \alpha D_S + H_3(w), r_i P)) \\
&= V_i
\end{aligned}$$

So, $H_4(\hat{e}(T_2' + SK_{S2} + T_3', U)) \neq V$.

- If $C(@)$ is generated for Eve, that means $T_w = [T_1, T_2, T_3]$ where $T_1 = \lambda P$, $T_2 = H_2(w) SK_{E2} \oplus \lambda PK_{S1}$ and $T_3 = H_3(w) \oplus \lambda PK_{S1}$. We can get

$$\begin{aligned}
&H_4(\hat{e}(T_2' + SK_{S2} + T_3', U)) \\
&= H_4(\hat{e}(H_2(w) \beta D_E + \alpha D_S + H_3(w), r_i P)) \\
&\neq H_4(\hat{e}(H_2(w) \beta D_R + \alpha D_S + H_3(w), r_i P)) \\
&= V_i
\end{aligned}$$

So, $H_4(\hat{e}(T_2' + SK_{S2} + T_3', U)) \neq V$.

By the proof of Theorem 1, the third property listed in section 3.1 is satisfied.

5.2 Security

Our construction relies on the intractable BDH

assumption for security. Certificateless cryptography is a branch of identity-based cryptography which is secure only in the random oracle model, and it is an open problem to build a secure identity-based encryption without random oracle. We still prove the security of our construction in the random oracle model by the following theorem.

Theorem 2(Security). The non-interactive CLPEKS scheme is semantically secure against a chosen keyword attack in the random oracle model assuming BDH problem is intractable.

Suppose $\mathcal{A}$ is an adversary of $\mathcal{A}_I$ and $\mathcal{A}_{II}$ that breaks CLPEKS with advantage ϵ, and makes q_{H_1} queries to H_1, q_{H_4} queries to H_4 and q_t trapdoor queries at most. q_{H_1}, q_{H_4} and q_t are positive.

We construct an algorithm $\mathcal{B}$ that solves the BDH problem with probability at least ϵ', and $\epsilon' \geq \lambda\epsilon$. Here, λ is a constant independent of ϵ. If BDH is intractable in G_1, ϵ' is a negligible function, and then ϵ is a negligible function too. Finally, we can conclude that the CLPEKS scheme is semantically secure against a chosen keyword attack in the random oracle model.

Firstly, we show the construction of the algorithm $\mathcal{B}$. Next, λ will be calculated through some claims for $\mathcal{A}_I$ and $\mathcal{A}_{II}$ respectively. Thus, Theorem 2 is proved based on Lemma 1 and Lemma 2.

Lemma 1. The non-interactive CLPEKS scheme is semantically secure against kind of adversary $\mathcal{A}_I$ in the random oracle model assuming BDH problem is intractable and $\epsilon' \geq \epsilon/(eq_{H_4}(1/(q_{H_1}+1) - 1/q_t + 1/(q_t q_{H_1} + q_t)))$.

Proof of Lemma 1.

Setup: Algorithm $\mathcal{B}$ starts by running **Setup** algorithm and generates public parameter *prm* and the *master-key* λ. Then, given $u_1=aP$, $u_2=bP$ and $u_3=cP$, Algorithm $\mathcal{B}$'s goal is to calculate $\hat{e}(P,P)^{abc}$. In the processing of interactions, we say that $\mathcal{B}$ is the challenger. Finally, u_1, denoted as $PK_{\mathcal{A}_I 2}$, and *prm* are send to adversary $\mathcal{A}_I$.

***H_1*-Query**: To response H_1 queries, an empty- initial list of tuples $< ID^{(j)}, h_1^{(j)}, coin_1^{(j)} >$ is maintained by $\mathcal{B}$, called *H_1-list*. When adversary makes a H_1 query with $ID_i \in \{0,1\}^*$, $\mathcal{B}$ response as follows:

1. If there exists a l satisfying a tuple $< ID^{(l)}, h_1^{(l)}, coin_1^{(l)} >$ is on the *H_1-list*, the matched tuple is send to adversary as response.

2. Otherwise, $\mathcal{B}$ picks up a $coin_1^{(i)} \in \{0,1\}$ such that $Pr[coin_1^{(i)} = 0] = 1/(q_{H_1}+1)$.

3. If $coin_1^{(i)} = 0$, let $e_1^{(i)} \in_R Z_q^*$. Otherwise $e_1^{(i)} = 1$. $\mathcal{B}$ sets $h_1^{(i)} = e_1^{(i)}$.

4. The tuple $< ID^{(i)}, h_1^{(i)}, coin_1^{(i)} >$ is added into *H_1-list* and send back as response.

***H_w*-Query:** To response H_2 and H_3 queries, an empty-initial list of tuples $< w^{(j)}, h_2^{(j)}, h_3^{(j)}, a^{(j)}, coin_w^{(j)} >$ is maintained by $\mathcal{B}$, called *H_w-list*. When adversary makes a H_2 or H_3 query with $w_i \in \{0,1\}^*$, $\mathcal{B}$ does the following steps to response:

1. If there exists a l satisfying a tuple $< w^{(l)}, h_2^{(l)}, h_3^{(l)}, a^{(l)}, coin_w^{(l)} >$ is on the *H_w-list*, $H_2(w_i) = h_2^{(l)}$ or $H_3(w_i) = h_3^{(l)}$ is send to adversary as response.

2. Otherwise, $\mathcal{B}$ picks up a $coin_w^{(i)} \in \{0,1\}$ such that $Pr[coin_w^{(i)} = 0] = 1/(q_{H_w}+1)$.

3. $\mathcal{B}$ picks up $a^{(i)} \in_R Z_q$ and $h_3^{(i)} \in_R G_1$. If $coin_w^{(i)} = 0$, then $h_2^{(i)} = a^{(i)}P + u_2$. Otherwise, $h_2^{(i)} = a^{(i)}Q_R$.

4. The tuple $< w^{(i)}, h_2^{(i)}, h_3^{(i)}, a^{(i)}, coin_w^{(i)} >$ is added into *H_w-list* and send back as response.

***H_4*-Query**: To response H_4 queries, similarly an empty-initial list of tuples $< t^{(j)}, V^{(j)}, coin_w^{(j)} >$ is maintained by $\mathcal{B}$, called *H_4-list*. When adversary makes a H_4 query with $t_i = h_4^{(i)} \in Z_q$, $\mathcal{B}$ does the following steps to response:

1. If there exists a l satisfying a tuple $< t^{(l)}, V^{(l)}, coin_w^{(l)} >$ is on the *H_4-list*, $H_4(t_i) = V^{(l)}$ is send to adversary as response.

2. Otherwise, do ***H_w*-Query** with t_i and get a tuple $< w_i, h_{2i}, h_{3i}, a_i, coin_{wi} >$ form *H_w-list*.

3. If $coin_w^{(i)} = 0$, $t_i = \hat{e}(a^{(i)}P + u_2, u_1))\hat{e}(Q_S, PK_{S2})\hat{e}(h_{3i}, P)$. Otherwise, $t_i = \hat{e}(a^{(i)}Q_R, u_1)\hat{e}(Q_S, PK_{S2})\hat{e}(h_{3i}, P)$.

4. Algorithm $\mathcal{B}$ chooses a $V_i \in_R \{0,1\}^f$. The tuple $< t_i, V_i, coin_{wi} >$ is added into *H_4-list* and send back as response.

Note that $h_1^{(i)}$, $h_2^{(i)}$, $h_3^{(i)}$ and $V^{(i)}$ are all indepen-

dent of adversary's current view as required.

The **Request** phase includes the 5 actions:

PPKExt: To response partial private key extraction with $ID_i \in \{0,1\}^*$, an empty-initial list of tuples $< ID^{(j)}, D^{(j)}, coin_1^{(j)} >$ is maintained by $\mathcal{B}$, called *D-list*. When adversary makes a **PPKExt** query with $ID_i \in Z_q$, $\mathcal{B}$ response as follows:

1. If there exists a l satisfying a tuple $< ID^{(l)}, D^{(l)}, coin_1^{(l)} >$ is on the *D-list*, the matched tuple is send to adversary as response.

2. Otherwise, do H_1**-Query** with ID_i and get a tuple $< ID_i, h_{1i}, coin_{1i} >$ from H_1*-list*.

3. If $coin_{1i} = 0$, $D_i = \lambda P$. and $D_i = P$ otherwise.

4. The tuple $< ID_i, D_i, coin_{1i} >$ is added into *D-list* and D is send back as response.

PriKGen: To response private key generation with $ID_i \in \{0,1\}^*$, an empty-initial list of tuples $< ID^{(j)}, x^{(j)}, SK^{(j)}, coin_1^{(j)} >$ is maintained by $\mathcal{B}$, called *SK-list*. When adversary makes such a query with $ID_i \in Z_q$, $\mathcal{B}$ does the following steps to response:

1. If there exists a l satisfying a tuple $< ID^{(l)}, x^{(l)}, SK^{(l)}, coin_1^{(l)} >$ is on the *SK-list*, the matched tuple is send to adversary as response.

2. Otherwise, do **PPKExt** query with ID_i and get a tuple $< ID_i, D_i, coin_{1i} >$ from *D-list*.

3. If $coin_{1i} = 0$, let x_i be a and be 1 otherwise.

4. Compute $SK_i = <SK_{i1}, SK_{i2}>$ where $SK_{i1} = x_i$ and $SK_{i1} = x_i D_i$.

5. The tuple $< ID_i, x_i, SK_i, coin_{1i} >$ is added into *SK-list* and send back as response.

PubKReq: To response public key request with $ID_i \in \{0,1\}^*$, $\mathcal{B}$ maintains an empty-initial list *PK-list* of tuples $< ID^{(j)}, PK^{(j)}, coin_1^{(j)} >$ and does the following steps to response:

1. If there exists a l satisfying a tuple $< ID^{(l)}, PK^{(l)}, coin_1^{(l)} >$ is on the *PK-list*, the matched tuple is send to adversary as response.

2. Otherwise, $\mathcal{B}$ does **PriKGen** query with ID_i and get a tuple $< ID_i, x_i, SK_i, coin_{1i} >$.

3. $PK_i = < PK_{i1}, PK_{i2} >$ where $PK_{i1} = x_i P$ and $PK_{i2} = x_i PK_C$.

4. The tuple $< ID_i, PK_i, coin_{1i} >$ is added into *PK-list* and send to adversary as response.

PKRep: To response public key replacement with $PK_{i'} = < PK_{i1'}, PK_{i2'} >$ for $ID_i \in \{0,1\}^*$, $\mathcal{B}$ does the following steps to response:

1. If there does not exist a l satisfying a tuple $< ID^{(l)}, PK^{(l)} >$ is on the *PK-list* and $ID_i = ID^{(l)}$, abort.

2. Do check whether $\hat{e}(PK_{i1'}, PK_C)) = \hat{e}(PK_{i2'}, P)$ or not. If not, abort.

3. Update *PK-list* with $< ID_i, PK_{i'} >$.

Trapdoor Query: When adversary makes a query for the trapdoor corresponding to the keyword w_i of ID_i, algorithm $\mathcal{B}$ responds as follows:

1. Do H_w query for w_i and get a tuple $<w_i, h_{2i}, h_{3i}, a_i, coin_{wi}>$ on which $H_2(w_i) = h_2^{(i)}$ and $H_3(w_i) = h_3^{(i)}$.

2. If $coin_{wi} = 0$, abort.

3. Otherwise, compute $T_i = a^{(i)} PK_{i1}$ so that T_i is the valid trapdoor for w_i under the public key u_1. T_i is send to adversary as response.

Challenge: At some time, $\mathcal{A}_I$ with ID^* will make challenge on w_0 and w_1 where $w_0 \neq w_1$. Then algorithm $\mathcal{B}$ generates the challenge $EP(w_b)$ as follows:

1. Do H_2 queries twice to obtain $h_2^{(i)}, h_3^{(i)}$ so that $H_2(w_i) = h_2^{(i)}$ and $H_3(w_3) = h_3^{(i)}$ where i=0,1. According to H_w*-list*, if both $coin_w^{(0)} = 1$ and $coin_w^{(1)} = 1$, abort.

2. Otherwise, $coin_w^{(0)} = 0$ or $coin_w^{(1)} = 0$ or $coin_w^{(0)} = coin_w^{(1)} = 0$.

3. Do **PubKReq** queries with ID^* to obtain PK_i. According to *PK-list*, if $coin_1^* = 1$, abort.

4. Otherwise, $coin_1^* = 0$.

5. Pick up a $b \in_R \{0,1\}$ such that $coin_w^{(b)} = 0$.

6. Pick up a $J \in_R \{0,1\}^f$ and sets $CP_{w_b} = [u_3, J]$. Notice that this challenge implicitly defines $[u_3, J]$ is a valid CLPEKS such that

$$J = H_4((\hat{e}(a^{(i)}P + u_2, u_1)\hat{e}(Q_S, PK_{S2})\hat{e}(h_{3i}, P))^c)$$
$$= H_4(\hat{e}(a^{(i)}P + bP, aP)^c \hat{e}(Q_S, PK_{S2})^c)\hat{e}(h_{3i}, P)^c)$$
$$= H_4(\hat{e}(P, P)^{\alpha c(a^{(i)}+b)}\hat{e}(SK_{S2}, PK_C)\hat{e}(h_{3i}, PK_C))$$

So, CP_{w_b} is valid for w_b as required.

More trapdoor queries: Continuously, $\mathcal{A}_I$ makes queries for the trapdoor corresponding to the word w_i where $w_i \neq w_0$ and $w_i \neq w_1$. $\mathcal{B}$ responds to these queries as before.

Attack: Finally, $\mathcal{A}_I$ makes a guess $b' \in \{0,1\}$ indicating whether CP_{w_b} is the challenge of CP_{w_0} or CP_{w_1}. Because $\mathcal{A}_I$ have made a trapdoor query for w_i or w_0, t exists on H_4-*list* with probability 1/2. If $\mathcal{B}$ picks the pair $<t,V>$, $\hat{e}(P,P)^{abc}$ can be computed as below:

$$\begin{aligned} & t/(\hat{e}(u_1,u_3)^{a^{(i)}}\hat{e}(SK_{S2},u_3)\hat{e}(h_{3i},u_3)) \\ &= \frac{\hat{e}(P,P)^{ac(a^{(i)}+b)}\hat{e}(SK_{S2},PK_C)\hat{e}(h_{3i},PK_C)}{\hat{e}(u_1,u_3)^{a^{(i)}}\hat{e}(SK_{S2},PK_C)\hat{e}(h_{3i},PK_C)} \\ &= \frac{\hat{e}(P,P)^{ac a^{(i)}}\hat{e}(P,P)^{abc}}{\hat{e}(P,P)^{ac a^{(i)}}} \\ &= \hat{e}(P,P)^{abc} \end{aligned} \quad (1)$$

For now, we construct an algorithm $\mathcal{B}$ that solves the BDH problem. Next, it needs to prove that $\mathcal{B}$ calculates $\hat{e}(P,P)^{abc}$ correctly with probability at least ϵ'. It must be satisfied that $\mathcal{B}$ does not abort during the game with a high probability. We define three events [17]:

E_1: $\mathcal{B}$ does not abort as a result of any of $\mathcal{A}$'s trapdoor queries.

E_2: $\mathcal{B}$ does not abort during the challenge phase.

E_3: $\mathcal{A}_I$ does not make a trapdoor query for either one of w_0 and $w1$.

Claim 1. *Pr[E_1]≥1/e.*

Proof of Claim 1. Suppose adversary does not make a trapdoor query of the same keyword twice. The probability that a trapdoor query causes $\mathcal{B}$ to abort is $1/(q_t+1)$. For a single query, suppose that the keyword w_i is the first element of the tuple $<w_i,h_{2i},h_{3i},a_i,coin_{wi}>$ on H_w-*list* and t_i is the first element of the tuple $<t_i,V_i,coin_{wi}>$ on H_4-*list*. t_i is the only element that depends on $coin_{wi}$. Therefore, the probability that this query causes $\mathcal{B}$ to abort is at most $1/(q_t+1)$. Since A makes at most q_t trapdoor queries, $Pr[E_1] = (1-1/(q_t+1))^{q_t} \geq 1/e$.

Claim 2. $Pr[E_2] \geq 1-1/(q_{H_1}+1)-1/q_t +1/(q_t q_{H_1}+q_t)$.

Proof of Claim 2. Algorithm $\mathcal{B}$ will abort during the challenge phase iff $coin_{w0} = coin_{w1} = coin_1^* = 1$. $coin_{w0}$, $coin_{w1}$ and $coin_1^*$ are independent of $\mathcal{A}_I$'s current view. Therefore, we can get $Pr[coin_{w0}=0]= Pr[coin_{w1}=0]= 1/(q_t+1)$ and $Pr[coin_1^*=0] = 1/(q_{H_1}+1)$. So, we will have that $Pr[coin_{w0}=coin_{w1}=1] = (1-1/(q_t+1))^2 \leq 1-1/q_t$ and $Pr[coin_1^*=1] = 1-1/(q_{H_1}+1)$. Because H_1 query is independent of H_w query,

$$\begin{aligned} Pr[E_2] &= (1-Pr[c_0=c_1=1])Pr[coin_1^*=1] \\ &\geq (1-1/q_t)(1-1/(q_{H_1}+1)) \\ &\geq 1-1/(q_{H_1}+1)-1/q_t+1/(q_t q_{H_1}+q_t). \end{aligned}$$

Both E_1 and E_2 are independent of each other. Therefore,

$$\begin{aligned} & Pr[E_1 \wedge E_2] \\ &= Pr[E_1]\cdot Pr[E_2] \\ &= (1/e)(1-1/(q_{H_1}+1)-1/q_t+1/(q_t q_{H_1}+q_t)) \\ &= (1/(q_{H_1}+1)-1/q_t+1/(q_t q_{H_1}+q_t))(1/e) \end{aligned}$$

Claim 3. $Pr[\neg E_3] \geq 2\epsilon$.

Proof of Claim 3. b is independent of $\mathcal{A}$'s current view. Therefore, $Pr[b'=b] \leq 1/2$. Because $\mathcal{A}$ is an adversary that breaks CLPEKS with advantage ϵ, $|Pr[b'=b]-1/2| \geq \epsilon$.

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

So, $\epsilon \leq Pr[\neg E_3]/2$.

Overall, when E_3 does not occur, $\mathcal{A}$ makes a trapdoor query for w_b and $EP(w_b)$ is valid with probability at least ϵ. That means w_b exists on H_4-*list*. Algorithm $\mathcal{B}$ will choose the correct pair with probability at least $1/q_{H_4}$. So,

$$\begin{aligned} \epsilon' &\geq \frac{1}{2}Pr[E_3]\cdot\frac{1}{q_{H_4}}Pr[E_1\wedge E_2] \\ &= \frac{1}{2}\cdot 2\epsilon\cdot\frac{1}{q_{H_4}}\cdot(1/(q_{H_1}+1)-1/q_t+1/(q_t q_{H_1}+q_t))(1/e) \\ &= \epsilon/(eq_{H_4}(1/(q_{H_1}+1)-1/q_t+1/(q_t q_{H_1}+q_t))). \end{aligned}$$

Lemma 2. The non-interactive CLPEKS scheme is semantically secure against kind of adversary $\mathcal{A}_{II}$ in the random oracle model assuming BDH problem is intractable and $\epsilon' \geq \epsilon/(eq_t q_{H_4})$.

Proof of Lemma 2.

Setup: Algorithm $\mathcal{B}$ starts by running **Setup** algorithm and generates public parameter *prm* and the *master-key* λ. u_1, λ and *prm* are send to the adversary $\mathcal{A}_{II}$.

***H_1*-Query:** To response H_1 queries, an empty-initial list of tuples $< ID^{(j)}, h_1^{(j)} >$ is maintained by $\mathcal{B}$, called H_1-*list*. When adversary makes a H_1 query with $ID_i \in \{0,1\}^*$, $\mathcal{B}$ does the following steps to response:

1. If there exists a l satisfying a tuple $< ID^{(l)}, h_1^{(l)} >$ is on the H_1-*list*, the matched tuple is send to adversary as response.

2. Pick up an $e_1^{(i)} \in_R Z_q$ and set $h_1^{(i)} = e_1^{(i)}$.

3. The tuple $< ID^{(i)}, h_1^{(i)} >$ is added into H_1-*list* and send back as response.

***H_w*-Query**: This oracle query is almost identical to that in proof of Lemma 1 and is omitted here.

***H_4*-Query:** This oracle query is almost identical to that in proof of Lemma 1 and is omitted here.

The **Request** phase includes following 4 actions:

PPKExt: To response partial private key extraction with $ID_i \in \{0,1\}^*$, an empty-initial list of tuples $< ID^{(j)}, D^{(j)} >$ is maintained by $\mathcal{B}$, called *D-list*. When adversary makes a **PPKExt** query with $ID_i \in Z_q$, $\mathcal{B}$ does the following steps to response:

1. If there exists a l satisfying a tuple $< ID^{(l)}, D^{(l)} >$ is on the *D-list*, the matched tuple is send to adversary as response.

2. Otherwise, do ***H_1*-Query** with ID_i and get a tuple $< ID_i, h_{1i} >$ from H_1-*list*.

3. Because $\mathcal{A}_{II}$ has access to *master-key*, compute $D_i = \lambda P$.

4. The tuple $< ID_i, D_i >$ is added into *D-list* and D_i is send back as response.

PriKGen: To response private key generation with $ID_i \in \{0,1\}^*$, an empty-initial list of tuples $< ID^{(j)}, x^{(j)}, SK^{(j)} >$ is maintained by $\mathcal{B}$, called *SK-list*. When adversary makes such a query with $ID_i \in Z_q$, $\mathcal{B}$ does the following steps to response:

1. If there exists a l satisfying a tuple $< ID^{(l)}, x^{(l)}, SK^{(l)} >$ is on the *SK-list*, the matched tuple is send to adversary as response.

2. Otherwise, do **PPKExt** query with ID_i and get a tuple $< ID_i, D_i >$ from *D-list*.

3. Let x_i be a, and compute $SK_i =< SK_{i1}, SK_{i2} >$ where $SK_{i1} = x_i$ and $SK_{i2} = x_i D_i$.

4. The tuple $< ID_i, x_i, SK_i >$ is added into *SK-list* and send back as response.

PubKReq: To response public key request with $ID_i \in \{0,1\}^*$, $\mathcal{B}$ maintains an empty-initial list *PK-list* of tuples $< ID^{(j)}, PK^{(j)} >$ and does the following steps to response:

1. If there exists a l satisfying a tuple $< ID^{(l)}, PK^{(l)} >$ is on the *PK-list*, the matched tuple is send to adversary as response.

2. Otherwise, $\mathcal{B}$ does **PriKGen** query with ID_i and get a tuple $< ID_i, x_i, SK_i >$.

3. $PK_i =< PK_{i1}, PK_{i2} >$ where $PK_{i1} = x_i P$ and $PK_{i2} = x_i PK_C$.

4. The tuple $< ID_i, PK_i >$ is added into *PK-list* and send back to adversary as response.

Trapdoor Query: This query is almost identical to that in proof of Lemma 1 and is omitted here.

Challenge: At some time, $\mathcal{A}_{II}$ with ID^* will make challenge on w_0 and w_1 where $w_0 \neq w_1$. Then algorithm $\mathcal{B}$ generates the challenge $EP(w_b)$ as follows:

1. Do H_2 queries twice to obtain $h_2^{(i)}, h_3^{(i)}$ so that $H_2(w_i) = h_2^{(i)}$ and $H_3(w_3) = h_3^{(i)}$ where i=0,1. According to H_w-*list*, if both $coin_w^{(0)} = 1$ and $coin_w^{(1)} = 1$, abort.

2. Otherwise, $coin_w^{(0)} = 0$ or $coin_w^{(1)} = 0$ or $coin_w^{(0)} = coin_w^{(1)} = 0$.

3. Do **PubKReq** queries with ID^* to obtain PK_i.

4. Pick up a $b \in_R \{0,1\}$ such that $coin_w^{(b)} = 0$.

5. Pick up a $J \in_R \{0,1\}^f$ and sets $CP_{w_b} = [u_3, J]$. Notice that this challenge implicitly defines $[u_3, J]$ is a valid CLPEKS too.

So, CP_{w_b} is valid for w_b as required.

More trapdoor queries: Continuously, $\mathcal{A}_{II}$ makes queries for the trapdoor corresponding to the word w_i where $w_i \neq w_0$ and $w_i \neq w_1$. $\mathcal{B}$ responds to these queries as before.

Attack: Finally, $\mathcal{A}_I$ makes a guess $b' \in \{0,1\}$ indicating whether CP_{w_b} is the challenge of CP_{w_0} or CP_{w_1}. Because $\mathcal{A}_I$ must make a trapdoor query for w_0 or w_1, t exists on H_4-*list* with probability 1/2. If $\mathcal{B}$ picks the pair $<t,V>$, $\hat{e}(P,P)^{abc}$ can be computed as described in Equation (1).

For now, we construct an algorithm $\mathcal{B}$ that solves the BDH problem. Next, it needs to prove that $\mathcal{B}$ calculates $\hat{e}(P,P)^{abc}$ correctly with probability at least ϵ'. It must be satisfied that $\mathcal{B}$ does not abort during the game with a high probability. We analyze three events defined in Lemma 1.

Claim 4. *$Pr[E_1] \geq 1/e$.*

Proof of Claim 4. This proof is almost identical to that of Claim 1 and is omitted

here.

Claim 5. $Pr[E_2] \geq 1/q_t$.

Proof of Claim 5. Algorithm $\mathcal{B}$ will abort during the challenge phase if and only if $coin_{w0} = coin_{w1} = coin_1^* = 1$. $coin_{w0}$ and $coin_{w1}$ are independent of $\mathcal{A}_{II}$'s current view. Therefore, we get $Pr[coin_{w0} = 0] = Pr[coin_{w1} = 0] = 1/(q_t + 1)$. So, we have that $Pr[coin_{w0} = coin_{w1} = 1] = (1 - 1/(q_t + 1))^2 \leq 1 - 1/q_t$. Hence, $Pr[E_2] \leq 1 - Pr[coin_{w0} = coin_{w1} = 1] \leq 1/q_t$.

Both E_1 and E_2 are independent of each other. Therefore, $Pr[E_1 \wedge E_2] = Pr[E_1] \cdot Pr[E_2] \geq 1/eq_t$.

Claim 6. $Pr[\neg E_3] \geq 2\epsilon$.

Proof of Claim 6. This proof of this claim is almost identical to that of Claim 3 and is omitted here.

Overall, when E_3 does not occur, $\mathcal{A}$ makes a trapdoor query for w_b and $EP(w_b)$ is valid with probability at least ϵ. That means w_b exists on H_4-*list*. Algorithm $\mathcal{B}$ will choose the correct pair with probability at least $1/q_{H_4}$. So,

$$\begin{aligned}\epsilon' &\geq \frac{1}{2} Pr[E_3] \cdot \frac{1}{q_{H_4}} Pr[E_1 \wedge E_2] \\ &= \frac{1}{2} \cdot 2\epsilon \cdot \frac{1}{q_{H_4}} \cdot \frac{1}{e} \cdot \frac{1}{q_t} \\ &= \epsilon / eq_t q_{H_4}.\end{aligned}$$

5.3 Efficiency

Search speed is the key characteristic for keyword search algorithm. In this section, we compare our construction with [2] and [3]. **Table I** shows a performance comparison in detail.

In the table, M is the multiplication operation in G_1, MU is the multiplication operation between G_1 and Z_r, E is the exponentiation operation, I is the invert in Z_r, and A is the addition in G_1. It is also clear that only the proposed CLPEKS both overcomes key escrow problem and provide trapdoor indistinguishable (TrapInd). In keyword search, search speed is a key characteristic to measure the efficiency. Based on pairings, our scheme is more efficient than Rhee's and as efficient as Baek's. Furthermore, our construction removes the key escrow problem which exists in Rhee's and Baek's.

Table I *Performance comparison*

	KeyGen	PEKS	Trapdoor
Baek's[2]	M	$E+M+2\hat{e}$	M
Rhee's[3]	$2E+I$	$2E+2\hat{e}$	$3E+I$
CLPEKS	$3M$	$MU+4M+3\hat{e}$	$3M$
	Test	Key Escrow	TrapInd
Baek's[2]	$M+\hat{e}$	Yes	No
Rhee's[3]	$3M+I+\hat{e}$	Yes	Yes
CLPEKS	$M+2A+\hat{e}$	No	Yes

VI. Experiments

For illustrating CLPEKS's superiority, massive experiments are conducted on a real Email dataset. Experiments show that the efficiency of CLPEKS is at least as same as that of the scheme in [3], but CLPEKS removes the heavy key escrow problem.

6.1 Environment

Enron[1] Email dataset, a famous Email dataset in information retrieval field, was collected and prepared by the CALO Project from 150 different users. Enron Email dataset contains 517,511 mails (duplication included) for 150 different users. 1,896,137 (duplication included) / 26,825 (duplication excluded) keywords are extracted from all the mails. Each mail contains about 3.7 keywords and each keyword appears about 70.7 times. We randomly select 10 different users for measuring efficiency of CLPEKS and reference scheme and 10 different keywords for each user. The detailed description of the selected mails for retrieval is shown in **Fig. 1**. *#(mail)* is the number of mails for each user, *#(keyword)* is the number of keywords for each user, and *Avg(keyword)* is the average number of keywords for each mail.

The Pairing-Based Cryptography (PBC) library[2] is adopted to perform the mathematical operations underlying pairing-based cryptosystems. All experiments are conducted on a

[1] http://www.cs.cmu.edu/~enron/

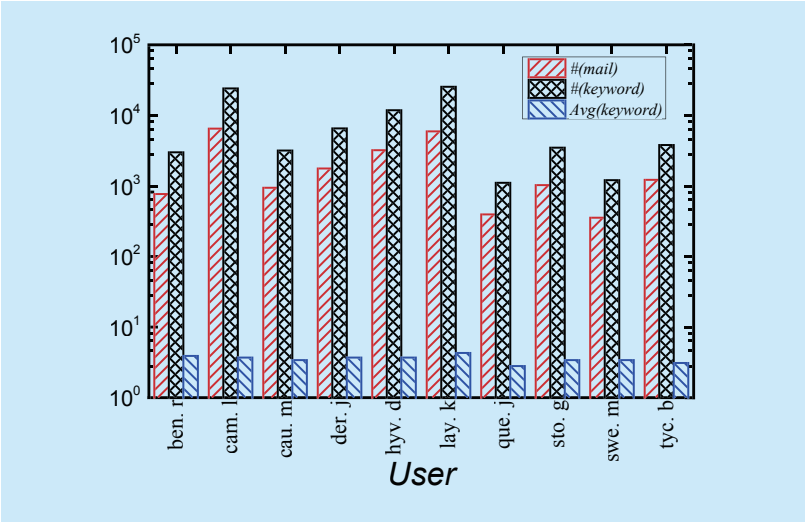

Fig.1 *Users and Mails for Retrieval*

Table II *Time Cost of Cryptographic Computation*

Operation	Time(μs)
Multiplication in G_1	3518.654
MUltiplication between G_1 and Z_r	15.837
Exponentiation	3516.353
Invert in Z_r	2.688
ê	4565.201
Addition in G_1	15.829

Linux machine with an Intel Core 2 Q9550 2.83GHz CPU and 4GB memory. We list the time cost of basic cryptographic computation in **Table II** through 1000 times repeat.

Through **Table I** and **Table II**, it is clear that *M*, *E*, and ê take over 1000 times time more than *MU*, *I*, and *A*. Hence, **Test** phase in our construction takes only about 50% time of Rhee's and as equal as Baek's.

For illustration purposes, the selected n users for retrieval are denoted as u_i, where $1 \le i \le n$. Here, $n=10$. Two key measures are employed to estimate the real performance of proposed CLPEKS:

- *Avg(i)* is used to evaluate the speed of retrieving mails for each mail of user. For k keyword w_j of each user, where $1 \le j \le k$, PEKS scheme takes t_j μs to returns mails containing the keywords. *Avg(i)* is defined as follows:

$$Avg(i) = \frac{1}{n\#(mail)_i} \sum_{j=1}^{k} t_j.$$

- *Time(i)* is used to evaluate the speed of retrieving mails for each user. For k keyword w_j of each user, where $1 \le j \le k$, PEKS scheme takes t_j μs to returns mails containing the keywords. *Time(i)* is defined as follows:

$$Time(i) = \frac{1}{n} \sum_{j=1}^{k} t_j.$$

6.2 Comparison

For illustrating CLPEKS's superiority visually, we compare CLPEKS with plain keyword search and PEKS proposed in [3]. We also introduce a reference experiment, in which no encryption algorithm is adopted and keyword search is conducted on plaintext, called Plain scheme. We do not compare with Baek's [2], because Baek's PEKS do not provide trapdoor indistinguishable which is indispensable.

Comparison on *Avg(i)*. We compare the *Avg(i)* of three schemes in **Fig. 2**. In general, the time for checking a mail increases in proportion of the average number of keywords in a single mail. CLPEKS performs as well as Rhee's but less efficient than Plain scheme. Obviously, encryption introduces much more cost than plain scheme in mail retrieval.

Comparison on *Time(i)*. We compare the time of searching a single keyword for each user in **Fig. 3**. Time cost increase with the number of mails for each user proportionally. In general, CLPEKS still performs as well as Rhee's but less efficient than Plain scheme. Time cost (≈10s) is acceptable when the number of mails for user is less than 1,000 because encryption will introduce more time cost. So, our construction is practical for small confidential institution.

As discussed in section 5.3, time cost in our construction's Test phase should much faster than Rhee's. However, it is opposite experiments completely. In experiments, the efficiency of our construction is as the same as Rhee's. We give a brief declaration here. In the implementation of Test phase, other than the ê operation, all operations have

[2] http://crypto.stanford.edu/pbc/

been preprocessed and stored when testing the first keyword. In our construction, only $H_3(\hat{e}(T_2' + SK_S + T_3', U)) = V$ is calculated after the first keyword. So does for Rhee's. Hence, more keywords a mail contains, the efficiency of our construction and Rhee's are more approximate. We still emphasize that, when each mail contains a single keyword and the number of mails for each user is small, our construction is more efficient.

VII. Conclusion

In this paper, we analysis the demands of CLPEKS, and give an encryption model and a secure model of CLPEKS. Through a novel mechanism, a CLPEKS construction is proposed which enhance the function of CLPKC. Our construction is proved secure against a chosen keyword attack in the random oracle model assuming BDH is intractable. Compared to Rhee's PEKS, our construction have higher efficiency, but remove the key escrow problem. On a real Email dataset, massive experiments are conducted to illustrate the superiority of CLPEKS.

CLPEKS is an SCF-PEKS scheme resisting outside keyword-guessing attack and semantic secure against adaptive chosen keyword attack. We improve the performance of retrieval process, and remove key escrow problem which Rhee's scheme does not. When the number of mails for user is less than 1,000, time cost in Test phase is less than 10s. And, when each mail contains a single keyword and the number of mails for each user is small, our construction is more efficient than (50% time of) Rhee's.

As is well-known, computation of bilinear pairing is more expensive than other cryptographic tools. We will focus on removing pairing computation in **Test** phase next. Also, more complicated operations, such as conjunct, aggregation, et al, will be considered.

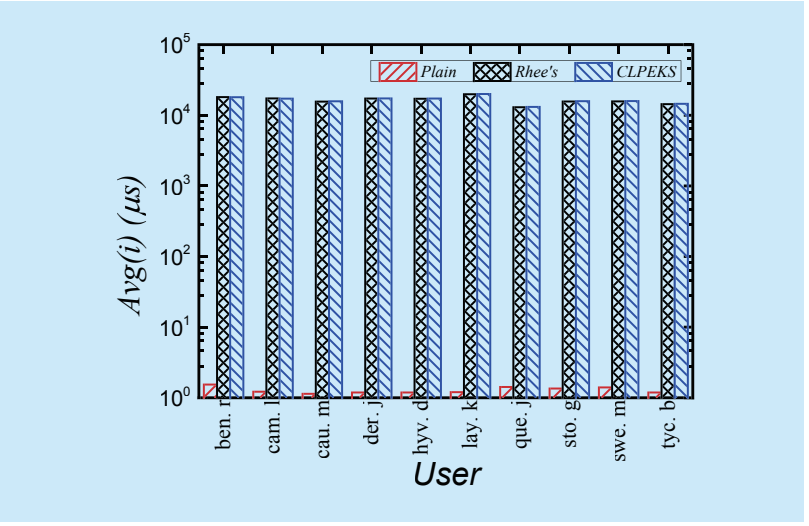

Fig.2 *Efficiency Comparison Between Schemes*

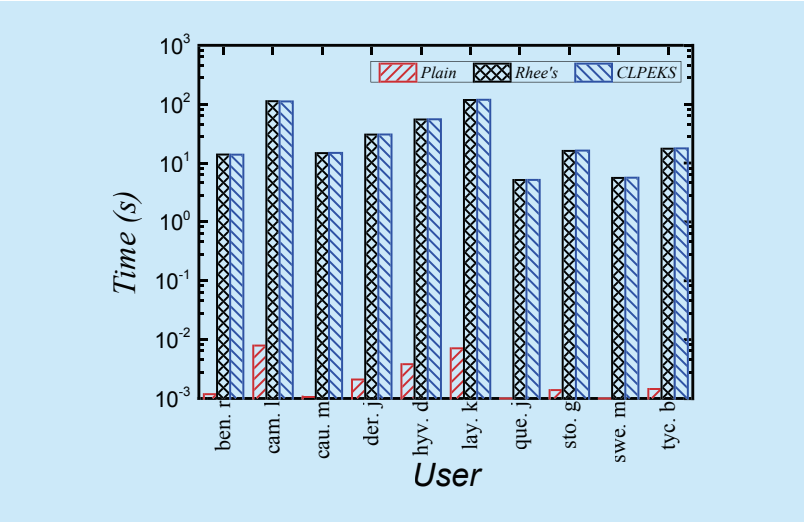

Fig.3 *Efficiency Comparison Between Schemes For Each Search*

Acknowledgements

This research was supported by the National Science Foundation of China for Funding Projects (61173089,61472298) and National Statistical Science Program of China(2013LZ46).

References

[1] SHAMIR A. Identity-Based Cryptosystems and Signature Schemes[C]// BLAKLEY G, CHAUM D. Advances in Cryptology. Lecture Notes in Computer Science, vol. 196. Berlin: Springer Berlin Heidelberg. 1985: 47-53.

[2] BAEK J, SAFAVI-NAINI R, SUSILO W. Public Key Encryption with Keyword Search Revisited[M]// GERVASI O, MURGANTE B, LAGAN A, et al. Computational Science and Its Applications ICCSA 2008. Lecture Notes in Computer Science, vol. 5072. Berlin: Springer Berlin / Heidel-

berg. 2008: 1249-1259.
[3] RHEE H S, PARK J H, SUSILO W, et al. Trapdoor security in a searchable public-key encryption scheme with a designated tester[J]. Journal of Systems and Software, 2010, 83(5): 763–771.
[4] RHEE H S, PARK J H, LEE D H. Generic construction of designated tester public-key encryption with keyword search[J]. Information Sciences, 2012, 205(0): 93–109.
[5] XU Peng, JIN Hai, WU Qianhong, et al. Public-Key Encryption with Fuzzy Keyword Search: A Provably Secure Scheme under Keyword Guessing Attack[J]. Computers, IEEE Transactions on, 2012, 62(11): 2266-2277.
[6] AL-RIYAMI S, PATERSON K. Certificateless Public Key Cryptography[M]// LAIH C S. Advances in Cryptology - ASIACRYPT 2003. Lecture Notes in Computer Science, vol. 2894. Berlin: Springer Berlin / Heidelberg. 2003: 452-473.
[7] DENT A. A survey of certificateless encryption schemes and security models[J]. International Journal of Information Security, 2008, 7(5): 349–377.
[8] BARBOSA M, FARSHIM P. Certificateless signcryption[C]// Proceedings of the 2008 ACM symposium on Information, computer and communications security. Tokyo, Japan: ACM. 2008: 369-372.
[9] WANG Fengjiao, ZHANG Yuqing. A new provably secure authentication and key agreement mechanism for SIP using certificateless public-key cryptography[J]. Computer Communications, 2008, 31(10): 2142–2149.
[10] CATALANO D, FIORE D, GENNARO R. Certificateless onion routing[C]// Proceedings of the 16th ACM conference on Computer and communications security. Chicago, Illinois, USA: ACM. 2009: 151-160.
[11] ZHANG Lei, ZHANG Futai, QIN Bo, et al. Provably-secure electronic cash based on certificateless partially-blind signatures[J]. Electronic Commerce Research and Applications, 2011, 10(5): 545–552.
[12] SEO S, NABEEL M, DING Xiaoyu, et al. An Efficient Certificateless Encryption for Secure Data Sharing in Public Clouds[J]. Knowledge and Data Engineering, IEEE Transactions on, 2013, 26(9): 2107-2119
[13] SONG D X, WAGNER D, PERRIG A. Practical techniques for searches on encrypted data[C]// Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on. 2000: 44-55.
[14] BONEH D, DI CRESCENZO G, OSTROVSKY R, et al. Public Key Encryption with Keyword Search[M]// CACHIN C, CAMENISCH J. Advances in Cryptology - EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027. Berlin: Springer Berlin / Heidelberg. 2004: 506-522.
[15] BYUN J, RHEE H, PARK H A, et al. Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data[M]// JONKER W, PETKOVI M. Secure Data Management. Lecture Notes in Computer Science, vol. 4165. Berlin: Springer Berlin Heidelberg. 2006: 75-83.
[16] BONEH D, LYNN B, SHACHAM H. Short Signatures from the Weil Pairing[C]// BOYD C. Advances in Cryptology ASIACRYPT 2001. Lecture Notes in Computer Science, vol. 2248. Berlin: Springer Berlin Heidelberg. 2001: 514-532.
[17] CORON J S. On the Exact Security of Full Domain Hash[C]// BELLARE M. Advances in Cryptology CRYPTO 2000. Lecture Notes in Computer Science, vol. 1880. Berlin: Springer Berlin Heidelberg. 2000: 229-235.
[18] QI Yanfeng,TANG Chunming,LOU Yu et al. Certificateless Proxy Identity-Based Signcryption Scheme Without Bilinear Pairings[J]. China Communications, 2013, 10(11): 37-41.

Biographies

***PENG Yanguo*,** is currently a Ph.D. student in Computer Architecture at Xidian University, Xi'an, China. His research interests include public key encryption with keyword search, high-dimensional secure search, data privacy protection.

***CUI Jiangtao*,** received the M.S. and Ph.D. degree both in Computer Science from Xidian University, Xi'an, China in 2001 and 2005 respectively. Between 2007 and 2008, he has been with the Data and Knowledge Engineering group working on high-dimensional indexing for large scale image retrieval, in the University of Queensland (Australia). He is currently a professor in School of Computer Science and Technology of Xidian University, China. His current research interests include data and knowledge engineering, and high-dimensional indexing.

***PENG Changgen*,** received his Ph.D. degree from Guizhou University, Guiyang, Guizhou, China. He is currently a professor at the College of Science in Guixzhou University. His current research interests include cryptography and information security. *The corresponding author. Email: gzu.cgpeng@gmail.com

***YING Zuobin*,** is currently a Ph.D. student in Computer Architecture at Xidian University, Xi'an, China. His research interests include identity-based cryptography and attribute-based cryptography.