Public-key Authenticate Encryption with Keyword Search Revised: Probabilistic TrapGen algorithm

Leixiao Cheng [1]*

[1]School of Mathematical Sciences, Fudan University, Shanghai 200433, China
fdu_clx@163.com

Abstract. Public key encryption with keyword search (PEKS) is first introduced by Boneh et al. enabling a cloud server to search on encrypted data without leaking any information of the keyword. In almost all PEKS schemes, the privacy of trapdoor is vulnerable to inside keyword guessing attacks (KGA), i.e., the server can generate the ciphertext by its own and then run the test algorithm to guess the keyword contained in the trapdoor. To sole this problem, Huang et al. proposed the public-key authenticated encryption with keyword search (PAEKS) achieving trapdoor privacy (TP) security, in which data sender not only encrypts the keyword but also authenticates it by using his/her secret key. Qin et al. introduced the notion of multi-ciphertext indistinguishability (MCI) security to capture outside chosen multi-ciphertext attacks, in which the adversary needs to distinguish two tuples of ciphertexts corresponding with two sets of keywords. They analysed that Huang's work cannot achieve MCI security, so they proposed an improved scheme to match both the MCI security and trapdoor privacy (TP) security. In addition, they also defined the notion of multi-trapdoor privacy (MTP) security, which requires to distinguish two tuples of trapdoors corresponding with two sets of keywords.
Unfortunately, trapdoor generation algorithms of all above works are deterministic, which means they are unable to capture the security requirement of MTP. How to achieve MTP security against inside multi-keyword guessing attacks,i.e., designing a probabilistic trapdoor generation algorithm, is still an open problem.
In this paper, we solve this problem. We initially propose two public-key authenticated encryption with keyword search schemes achieving both MCI security and MTP security simultaneously. We provide formal proof of our schemes in the random oracle model.

Keywords: Public key encryption · Keyword search · Keyword guessing attacks · Multi-ciphertext indistinguishability · Multi-trapdoor privacy

1 Introduction

Cloud computing is a promising computing paradigm, in which data storage and procession is shifted from terminal devices to the cloud, enabling users to

* Corresponding author

remotely maintain and manage data at lower costs. However, the data owner will lose his/her full control over data once the data is uploaded to the cloud. In this case, sensitive information is exposed to the cloud or other users. To protect the privacy of data, a straightforward method is encrypt the data before uploading it to the cloud server. Unfortunately, standard encryption technique doesn't provide search capability on the encrypted data.

In 2004, Boneh et al. [1] proposed the first public key encryption with keyword search (PEKS). In a PEKS scheme, data sender encrypts the data with a ciphertext keyword under the receiver's public key and uploads the ciphertext to the cloud. Searching for the ciphertext with specific keyword, the receiver generates the trapdoor with a target keyword and submits the trapdoor to the cloud. By interacting the trapdoor and each ciphertext, the cloud tests whether the target keyword of the trapdoor is identical to the ciphertext keyword of each ciphertext. If so, the cloud returns the successfully matched ciphertexts to the receiver. During the entire procedure, the cloud server learns nothing about the keywords embedded in the ciphertext and the trapdoor.

Attacks for PEKS are generally divided into two types: outside attack by malicious user and inside attacks by malicious tester (i.e., cloud server). The keyword space is ideally assumed to be super-polynomial, whereas in the real applications, keywords are often chosen from a low-entropy keyword space. Therefore, the privacy of trapdoor is compromised, since it's feasible for the adversary to guess the keywords containing in a given trapdoor by the keyword guessing attacks (KGA) [3]. Specifically, for each possible target keyword contained in the given trapdoor, the adversary encrypts it and generates a corresponding ciphertext, then checks whether it can be matched by the trapdoor. If succeed, the adversary knows that the ciphertext containing the same keyword as that of the given trapdoor. In the case of outside keyword guessing attacks (OKGA), any outside adversary obtained the given trapdoor, it can launch a KGA attack by doing the test freely. To resist OKGA, one can transmit the trapdoor by a secure channel between the receiver and the server so that only the cloud server can get the trapdoor; or restrict that only the designated server can do the test [15], i.e., any other unauthorized entity cannot check whether the trapdoor matches a ciphertext. However, neither method can prevent inside keyword guessing attacks (IGKA), since the attacks are launching by the server itself. Therefore, it's very necessary look for a method to achieve trapdoor privacy (TP) security against IKGA in a PEKS scheme.

Recently, Huang and Li [9] introduced the notion of Public-key Authenticated Encryption with Keyword Search (PAEKS) to resist IKGA, in which the sender encrypts a keyword under his public key and the senders secret key and then authenticates it, such that the cloud server cannot launch IKGA successfully by encrypting a keyword itself. Qin et al., [13] proposed a new security model for PAEKS, i.e., the multi-ciphertext indistinguishability (MCI), to achieve semantic security against outside chosen multi-keyword attacks. The MCI captures a practical attack on finding relations between two encrypted files containing several same keywords. They analysed that the scheme of [9] is secure against outside chosen single-keyword attacks but is vulnerable to chosen multi-keyword attacks. Thus they made slight but nontrivial modification to [1] to prevent both inside keyword guessing attack and outside chosen multi-keyword attack.

1.1 Motivation

In [13], besides MCI, they also defined the security model of multi-trapdoor privacy (MTP), to achieve semantic security against inside multi-keyword guessing attacks. The MTP captures a practical attack on finding relations between two trapdoors containing some same keywords. However, they didn't provide a concrete construction to achieve MTP security. Since the MTP requires the adversary to distinguish two tuples of trapdoors generated by two sets of keywords, to achieve MTP security, the trapdoor generation algorithm must be probabilistic. Otherwise, the adversary can easily distinguish the tuples by the difference of frequency of a keyword in the two keyword sets. This's the key point that [1, 8–11, 13] cannot achieve MTP security, since their trapdoor generation algorithms are all non-probabilistic. How to achieve MTP security against inside multi-keyword guessing attacks is still an open problem.

1.2 Our contributions

Motivated by the above observations, we initially propose two public-key authenticated encryption with keyword search (PAEKS) schemes achieving both multi-ciphertext indistinguishability (MCI) and multi-trapdoor privacy (MTP) simultaneously. Briefly, we make the following contributions in this paper:

1. At first, we solve the open problem proposed in [13], provide a concrete PAEKS scheme with probabilistic trapdoor generation algorithm to prevent both inside multi-keyword guessing attack and outside chosen multi-keyword attack. The security of our first scheme is proved under static assumptions (i.e., DDH assumption and modified DLIN assumption) in the random oracle model.
2. Then, to guarantee the integrity and validity of keys of sender and receiver, we provide our second PAEKS scheme with simplified key management. The security of second first scheme is proved under static assumptions (i.e., DBDH assumption) in the random oracle model.

In Table 1, we compare our schemes with some other PEKS schemes.

1.3 Related works

In 2000 [1], Song et al. [18] introduced the first searchable symmetric encryption (SSE), which allows keyword search over encrypted data in the outsourcing scenarios. In 2004, Boneh et al. [1] initialized proposed the first public key encryption with keyword search scheme (PEKS). In 2005, Abdalla et al. [1] revised Boneh's work and provided a transform from an anonymous identity-based encryption scheme to a secure PEKS scheme. In addition, they also extended the basic notions of anonymous hierarchical identity-based encryption, public-key

Table 1. Comparison with other PEKS schemes

	CI	MCI	TP	MTP	Probabilistic TrapGen
[1]	√	√			
[9]	√		√		
[11]	√		√		
[13]	√	√	√		
[10]	√	√	√		
Ours 1	√	√	√	√	√
Ours 2	√	√	√	√	√

CI: ciphertext indistinguishability against outside chosen single-ciphertext attacks.
MCI: multi-ciphertext indistinguishability against outside chosen multi-ciphertext attacks.
TP: trapdoor privacy against inside single-keyword guessing attacks.
MTP: multi-trapdoor privacy against inside multi-keyword guessing attacks.
Probabilistic TrapGen: the trapdoor generation algorithm in the scheme is probabilistic.

encryption with temporary keyword search, and identity-based encryption with keyword search. Since then, various PEKS schemes have been proposed. Golle et al. [7] defined the security model for conjunctive keyword search over encrypted data and presented the first schemes for conducting such searches securely. Park et al. [12] proposed the public key encryption with conjunctive field keyword search enabling an email gateway to search keywords conjunctively. Boneh et al. [2] provided several public key system that support comparison queries, subset queries and arbitrary conjunctive queries on encrypted data. Shi et al. [17] designed an encryption scheme called multi-dimensional range query over encrypted data to deal with the privacy issues related to the sharing of network audit logs. Moreover, proxy re-encryption with keyword search [16], decryptable searchable encryption [6], keyword updatable PEKS [14] and attribute-based encryption with keyword search [5, 19] enjoyed some other interesting features compared with standard PEKS.

However, in actual practice, keywords are chosen from much smaller space than the space of passwords, thus all the above schemes are susceptible to the keyword guessing attacks (KGA). Byun et al. [3] analysed the security vulnerabilities on [1, 7] by performing off-line keyword guessing attacks. Rhee et al. [15] introduced the concept of trapdoor indistinguishability and show that trapdoor indistinguishability is sufficient for thwarting keyword-guessing attacks and proposed a provably secure PEKS scheme in the designated tester model (d-PEKS). Unfortunately, dPEKS suffers from an inherent insecurity called inside keyword guessing attack (IKGA) launched by the malicious tester (i.e., cloud server). Chen et al. [4] proposed a new PEKS framework named dual-server PEKS scheme (DS-PEKS) using two uncolluded semi-trusted servers. Huang et al. [9] proposed a public-key authenticated encryption with keyword search scheme (PAEKS), which is secure against IKGA. It was then extended to certificateless PAEKS [8], identity-based setting with designed tester [10]. Recently, Noroozi et al. [11] found that Huang's work is insecure in the multi-receiver model and Qin et al. [13] extended the security model of PAEKS and proposed a PAEKS scheme secure against both inside keyword attacks and chosen multi-keyword attacks.

1.4 Organization

This paper is organized as follows. Section 2 describes the necessary preliminaries. Section 3 presents the system and security model. We give a concrete construction and explicit analysis of our PAEKS in section 4 and section 5 respectively. In the end, section 6 summarizes the paper and prospects for the future research.

2 Preliminaries

In this section, we introduce some background knowledge, which includes bilinear maps, Diffie-Hellman assumption and its variants.

2.1 Bilinear map

We briefly recall the definitions of the bilinear map. Let $\mathbb{G}_0$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a efficient computable bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$. The bilinear map e has a few properties: (1) Bilinearity: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$. (2) Non-degeneracy: for any generator $g \in \mathbb{G}_0$, $e(g, g) \in \mathbb{G}_T$ is the generator of $\mathbb{G}_T$. (3) Computability: For any $g, h \in \mathbb{G}_0$, there is an efficient algorithm to compute $e(g, h)$.

2.2 DDH assumption

The Diffie-Hellman (DDH) assumption is defined as: given group parameter $(\mathbb{G}_0, p, g)$ and three random elements $x, y \in_R \mathbb{Z}_p^*$. We say that the DDH assumption holds, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between the tuple (g, g^x, g^y, g^{xy}) and the tuple (g, g^x, g^y, ϑ), where ϑ is randomly selected from $\mathbb{G}_0$. More specifically, the advantage ϵ of $\mathcal{B}$ in solving the DDH problem is defined as

$$\left| \Pr[\mathcal{A}(g, g^x, g^y, g^{xy}) = 1] - \Pr[\mathcal{A}(g, g^x, g^y, \vartheta) = 1] \right|. \quad (1)$$

Definition 1 (DDH). *We say that the DDH assumption holds if no PPT algorithm has a non-negligible advantage ϵ in solving DDH problem.*

2.3 DBDH assumption

The decisional bilinear Diffie-Hellman (DBDH) assumption is defined as: given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$ and three random elements $x, y, z \in_R \mathbb{Z}_p^*$. We say that the DBDH assumption holds, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between the tuple $(g, g^x, g^y, g^z, e(g,g)^{xyz})$ and the tuple $(g, g^x, g^y, g^z, \vartheta)$, where ϑ is randomly selected from $\mathbb{G}_T$. More specifically, the advantage ϵ of $\mathcal{B}$ in solving the DBDH problem is defined as

$$\Big|\Pr[\mathcal{A}(g,g^x,g^y,g^z,e(g,g)^{xyz})=1]-\Pr[\mathcal{A}(g,g^x,g^y,g^z,\vartheta)=1]\Big|. \tag{2}$$

Definition 2 (DBDH). *We say that the DBDH assumption holds if no PPT algorithm has a non-negligible advantage ϵ in solving DBDH problem.*

2.4 mDLIN assumption

The Decisional Linear (DLIN) assumption is defined as: given $g, g^a, g^b, g^{ac}, g^{bd} \in \mathbb{G}_0$ and $a, b, c, d \in_R \mathbb{Z}_p^*$, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between the tuple $(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and the tuple $(g, g^a, g^b, g^{ac}, g^{bd}, \vartheta)$, where $\vartheta \in_R \mathbb{G}_0$, then the DLIN assumption holds.

In this paper, we make use of the variant of DLIN assumption called the modified DLIN assumption (mDLIN) [9]: given $g, g^a, g^b, g^{ac}, g^{d/b} \in \mathbb{G}_0$, the mDLIN assumption holds, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between the tuple $(g, g^a, g^b, g^{ac}, g^{d/b}, g^{c+d})$ and the tuple $(g, g^a, g^b, g^{ac}, g^{d/b}, \vartheta)$, where $\vartheta \in_R \mathbb{G}_0$. Specifically, the advantage ϵ of $\mathcal{B}$ in solving the DLIN problem is defined as

$$\Big|\Pr[\mathcal{A}(g,g^a,g^b,g^{ac},g^{d/b},g^{c+d})=1]-\Pr[\mathcal{A}(g,g^a,g^b,g^{ac},g^{d/b},\vartheta)=1]\Big|. \tag{3}$$

Definition 3 (mDLIN). *We say that the mDLIN assumption holds if no PPT algorithm has a non-negligible advantage ϵ in solving mDLIN problem.*

3 Definition and Security Model of PAEKS

Public-key authenticated encryption with keyword search (PAEKS) was first introduced by Huang et al. [9], in which the privacy of trapdoor is secure against inside keyword guessing attacks. In this section, we recall the definition and security model of PAEKS.

3.1 Definition of PAEKS

The PAEKS includes the following six algorithms:

- **Setup**$(1^\lambda) \rightarrow PP$: Given security parameter λ, the algorithm generates the global public parameter PP.

- **KeyGen**$_R(PP) \rightarrow (PK_R, SK_R)$: On input the public parameter PP, the algorithm generates the public/secret key pair (PK_R, SK_R) for the receiver.
- **KeyGen**$_S(PP) \rightarrow (PK_S, SK_S)$: On input the public parameter PP, the algorithm generates the public/secret key pair (PK_S, SK_S) for the data sender.
- **PAEKS**$(PP, PK_R, PK_S, SK_S, KW) \rightarrow CT$: On input public parameter PP, public key of receiver PK_R, the key pair of sender (PK_S, SK_S) and a ciphertext keyword KW, the algorithm generates the ciphertext CT related to the keyword KW.
- **TrapGen**$(PP, PK_R, PK_S, SK_R, KW') \rightarrow Tr$: On input public parameter PP, public key of receiver PK_S, the key pair of receiver (PK_R, SK_R) and a target keyword KW, this algorithm generates the trapdoor Tr of KW'.
- **Test**$(PP, Ct, Tr) \rightarrow 0/1$: On input public parameter PP, ciphertext CT and trapdoor Tr, this algorithm checks whether the ciphertext keyword KW is identical to the target keyword KW'. If so, it outputs 1; otherwise it outputs 0.

Correctness: for any honestly generated key pairs (PK_R,SK_R) and (PK_S,SK_S), any two keywords KW, KW', let PAEKS$(PP, PK_R, PK_S, SK_S, KW) \rightarrow CT$ and TrapGen$(PP, PK_R, PK_S, SK_R, KW') \rightarrow Tr$. If $KW = KW'$, then the text algorithm outputs 1 with probability 1 i.e., $\Pr[\text{Test}(PP, Ct, Tr) = 1] = 1$; otherwise $\Pr[\text{Test}(PP, Ct, Tr) = 0] = 1 - negl(\lambda)$.

3.2 Security Model

In this section, we describe two security model introduced by Qin et al. [13]: multi-ciphertext indistinguishability (MCI) security to capture outside chosen multi-ciphertext attacks and multi-trapdoor privacy (MTP) security to capture inside multi-keyword guessing attacks. The notion of MCI security in multi-challenge setting is defined as follows:

- **Setup**: Given a security parameter λ, the challenger $\mathcal{C}$ runs the algorithms Setup, KeyGen$_R$, KeyGen$_S$, and responds the adversary $\mathcal{A}$ with PP, PK_R, PK_S.
- **Phase** 1: The adversary $\mathcal{A}$ is allowed to issue polynomial queries the following oracles:

  Ciphertext Oracle $\mathcal{O}_C$**:** Given a ciphertext query (PK_R, PK_S, KW), $\mathcal{C}$ responds $\mathcal{A}$ with the ciphertext CT of keyword KW.

  Trapdoor Oracle $\mathcal{O}_T$**:** Given a trapdoor query (PK_R, PK_S, KW'), $\mathcal{C}$ responds $\mathcal{A}$ with the trapdoor Tr of keyword KW'.
- **Challenge**: $\mathcal{A}$ chooses two keyword sets $\{KW_0^{i*}\}_{i\in[1,I]}$ and $\{KW_1^{i*}\}_{i\in[1,I]}$ with the restriction that no element of $\{PK_R, PK_S, KW_j^{i*}\}_{j\in 0,1, i\in[1,I]}$ has been queried on $\mathcal{O}_C$ or $\mathcal{O}_T$. $\mathcal{C}$ randomly chooses a bit $\theta \in 0, 1$ and responds $\mathcal{A}$ with $\{CT_\theta^{i*}\}_{i\in[1,I]}$, where $CT_\theta^{i*} \leftarrow \text{PAEKS}(PP, PK_R, PK_S, SK_S)$.
- **Phase** 2: This phase is the same as Phase 1 with the restriction that no element of $\{PK_R, PK_S, KW_j^{i*}\}_{j\in 0,1, i\in[1,I]}$ can be queried on $\mathcal{O}_C$ or $\mathcal{O}_T$.
- **Guess**: $\mathcal{A}$ outputs a guess bit θ' of θ and it wins the game if $\theta' = \theta$. The advantage of $\mathcal{A}$ to win the MCI security game is defined as $Adv_{\mathcal{A}}^{MCI}(\lambda) = \left| \Pr[\theta' = \theta] - \frac{1}{2} \right|$.

Definition 4. *The PAEKS achieves MCI security against outside chosen multi-ciphertext attacks, if there exist no PPT adversary winning the above security game with a non-negligible advantage ϵ.*

To prevent inside keyword guessing attacks, Huang et al. [9] introduced the notion of single trapdoor privacy, which is extended to multi-trapdoor privacy (MTP) by Qin et al. [13]. The notion of MTP security in multi-challenge setting is defined as follows:

- **Setup**: Given a security parameter λ, the challenger $\mathcal{C}$ runs the algorithms $\text{Setup}, \text{KeyGen}_R, \text{KeyGen}_S$, and responds the adversary $\mathcal{A}$ with PP, PK_R, PK_S.
- **Phase** 1: The adversary $\mathcal{A}$ is allowed to issues polynomial queries the following oracles:

  Ciphertext Oracle $\mathcal{O}_C$: Given a ciphertext query (PK_R, PK_S, KW), $\mathcal{C}$ responds $\mathcal{A}$ with the ciphertext CT of keyword KW.

  Trapdoor Oracle $\mathcal{O}_T$: Given a trapdoor query (PK_R, PK_S, KW'), $\mathcal{C}$ responds $\mathcal{A}$ with the trapdoor Tr of keyword KW'.
- **Challenge**: $\mathcal{A}$ chooses two keyword sets $\{KW_0^{i*}\}_{i\in[1,I]}$ and $\{KW_1^{i*}\}_{i\in[1,I]}$ with the restriction that no element of $\{PK_R, PK_S, KW_j^{i*}\}_{j\in 0,1, i\in[1,I]}$ has been queried on $\mathcal{O}_C$ or $\mathcal{O}_T$. $\mathcal{C}$ randomly chooses a bit $\theta \in 0, 1$ and responds $\mathcal{A}$ with $\{Tr_\theta^{i*}\}_{i\in[1,I]}$, where $Tr_\theta^{i*} \leftarrow \text{TrapGen}(PP, PK_R, PK_S, SK_R)$.
- **Phase** 2: This phase is the same as Phase 1 with the restriction that no element of $\{PK_R, PK_S, KW_j^{i*}\}_{j\in 0,1, i\in[1,I]}$ can be queried on $\mathcal{O}_C$ or $\mathcal{O}_T$.
- **Guess**: $\mathcal{A}$ outputs a guess bit θ' of θ and it wins the game if $\theta' = \theta$. The advantage of $\mathcal{A}$ to win the MTP security game is defined as $Adv_{\mathcal{A}}^{MTP}(\lambda) = \left|\Pr[\theta' = \theta] - \frac{1}{2}\right|$.

Definition 5. *The PAEKS achieves MTP security against inside multi-keyword guessing attacks, if there exist no PPT adversary winning the above security game with a non-negligible advantage ϵ.*

4 PAEKS with both MCI and MTP

In this section, we propose our first PAEKS scheme that achieves both MCI and MTP securities simultaneously.

4.1 Construction

The construction detail of our first scheme is shown as follows.

- **Setup**$(1^\lambda, \mathcal{L}) \rightarrow (PP)$: Given a security parameter λ, the algorithm chooses a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_0$ and $\mathbb{G}_T$ are groups with prime order p and g is the generator of $\mathbb{G}_0$, and a hash function $H : \{0,1\}^* \longrightarrow \mathbb{G}_0$. It outputs the public parameter $PP = (e, \mathbb{G}_0, \mathbb{G}_T, p, g, H)$.
- $\mathbf{KeyGen_R}(PP) \rightarrow (PK_R, SK_R)$: The receiver picks a random $x \in_R \mathbb{Z}_p^*$ and sets $PK_R = g^x, SK_R = x$.

- **KeyGen$_\mathbf{S}$**$(PP) \rightarrow (PK_S, SK_S)$: The sender picks a random $y \in_R \mathbb{Z}_p^*$ and sets $PK_S = g^y, SK_S = y$.
- **PAEKS**$(PP, PK_R, PK_S, SK_S, KW) \rightarrow CT$: It randomly chooses $s, r \in_R \mathbb{Z}_p^*$, then computes

$$C_1 = H(PK_R, PK_S, KW)^{ys} g^r, C_2 = g^{xr}, C_3 = g^{ys}. \tag{4}$$

  Finally, it outputs $CT = (C_1, C_2, C_3)$.
- **TrapGen**$(PP, PK_R, PK_S, SK_R, KW') \rightarrow Tr$: It randomly chooses $r' \in_R \mathbb{Z}_p^*$, then computes

$$T_1 = H(PK_R, PK_S, KW')^{xr'}, T_2 = g^{xr'}, T_3 = g^{r'}. \tag{5}$$

  Finally, it outputs $Tr = (T_1, T_2, T_3)$.
- **Test**(PP, Ct, Tr): The algorithm checks whether the following equation holds or not:

$$e(C_1, T_2) = e(T_1, C_3) \cdot e(T_3, C_2). \tag{6}$$

  If so, it outputs 1; otherwise it outputs 0.

4.2 Correctness

The receiver's key pair is $(PK_R = g^x, SK_R = x)$ and the sender's key pair is $(PK_S = g^y, SK_S = y)$, while H is a collusion resistant hash function. If the ciphertext keyword KW is identical to the target keyword KW', we have the following equation:

$$\begin{aligned} e(C_1, T_2) =& e(H(PK_R, PK_S, KW)^{ys} g^r, g^{xr'}) \\ =& e(H(PK_R, PK_S, KW), g)^{xysr'} \cdot e(g, g)^{xrr'} \\ =& e(H(PK_R, PK_S, KW')^{xr'}, g^{ys}) \cdot e(g^{r'}, g^{xr}) \\ =& e(T_1, C_3) \cdot e(T_3, C_2). \end{aligned} \tag{7}$$

The above equation holds with probability 1, if $KW = KW'$; it doesn't hold with overwhelming probability when $KW \neq KW'$.

4.3 Security proof

In this section, we provide a security analysis of our scheme.

Theorem 1. *Our scheme is semantically MCI secure against outside chosen keyword attacks in the random oracle model under the mDLIN assumption.*

Proof. Supposed that a PPT adversary $\mathcal{A}$ can break the MCI security of our scheme with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a mDLIN tuple from a random one with a non-negligible probability. The mDLIN challenger $\mathcal{C}$ selects $a, b, c, d \in_R \mathbb{Z}_p^*$, $\theta \in \{0, 1\}$, $\mathcal{R} \in_R \mathbb{G}_0$ at random. Let $\mathcal{Z} = g^{c+d}$, if $\theta = 0$, $\mathcal{R}$ else. Next, $\mathcal{C}$ sends $\mathcal{B}$ the mDLIN tuple $\langle g, g^a, g^b, g^{ac}, g^{d/b}, \mathcal{Z} \rangle$. Then, $\mathcal{B}$ plays the role of simulator in the following security game.

- **Setup**: $\mathcal{B}$ selects a collusion resistant hash function H and sets $PK_R = g^a, PK_S = g^b$ which implies $SK_R = a, SK_S = b$. Then, it sends $PP = (e, \mathbb{G}_0, \mathbb{G}_T, p, g, H)$ and PK_R, PK_S to $\mathcal{A}$.
- **Phase1**: The adversary $\mathcal{A}$ may issue at most q_H, q_C, q_T queries to the hash oracle $\mathcal{O}_H$, the ciphertext oracle $\mathcal{O}_C$ and the trapdoor oracle $\mathcal{O}_T$. We make some necessary restriction: q_H, q_C, q_T are polynomial; $\mathcal{A}$ isn't allowed to repeat a query to the hash oracle $\mathcal{O}_H$ and may repeat r_C queries to $\mathcal{O}_C$ and r_T queries to $\mathcal{O}_T$; $\mathcal{A}$ couldn't issue a ciphertext query (PK_R, PK_S, KW) to $\mathcal{O}_C$ or a trapdoor query to $\mathcal{O}_T$ before issuing the hash query (PK_R, PK_S, KW) to $\mathcal{O}_H$.

  $\mathcal{B}$ simulates these oracles as follows:

  Hash Oracle $\mathcal{O}_H$: To response the hash query, $\mathcal{B}$ maintains a list of tuple $\{(PK_R, PK_S, KW_j), h_j, a_j, c_j\}$ called the "H-list". This list is initially empty. Given a hash query tuple (PK_R, PK_S, KW_i), $\mathcal{B}$ responds as follows:
  1. If (PK_R, PK_S, KW_i) has already existed in the H-list, $\mathcal{B}$ responds $\mathcal{A}$ with $H(PK_R, PK_S, KW_i) = h_i \in \mathbb{G}_0$.
  2. Otherwise, $\mathcal{B}$ flips a random coin $c_i \in \{0, 1\}$ with the probability $\Pr[c_i = 0] = \delta$.
  3. $\mathcal{B}$ picks a random $a_i \in_R \mathbb{Z}_p^*$. If $c_i = 0$, it sets
  $$H(PK_R, PK_S, KW_i) = h_i = g^{d/b} \cdot g^{a_i} \in \mathbb{G}_0;$$
  If $c_i = 1$, it sets
  $$H(PK_R, PK_S, KW_i) = h_i = g^{a_i} \in \mathbb{G}_0$$
  .
  4. $\mathcal{B}$ adds the tuple $\{(PK_R, PK_S, KW_j), h_j, a_j, c_j\}$ into the H-list and responds $\mathcal{A}$ with $H(PK_R, PK_S, KW_i) = h_i$.

  Ciphertext Oracle $\mathcal{O}_C$: Given a ciphertext query (PK_R, PK_S, KW_i), $\mathcal{B}$ retrieves the tuple $\{(PK_R, PK_S, KW_j), h_j, a_j, c_j\}$ from H-list. If $c_i = 0$, it aborts the game and outputs a random bit θ' as the guess of θ. Otherwise, it picks $s, r \in_R \mathbb{Z}_p^*$, and responds $\mathcal{A}$ with the ciphertext $Ct = (C_1, C_2, C_3)$, where
  $$C_1 = h_i^{bs} \cdot g^r = (g^b)^{a_i s} \cdot g^r, C_2 = (g^a)^r, C_3 = (g^b)^s.$$

  Trapdoor Oracle $\mathcal{O}_T$: Given a trapdoor query (PK_R, PK_S, KW_i), $\mathcal{B}$ retrieves the tuple $\{(PK_R, PK_S, KW_j), h_j, a_j, c_j\}$ from H-list. If $c_i = 0$, it aborts the game and outputs a random bit θ' as the guess of θ. Otherwise, it picks $r' \in_R \mathbb{Z}_p^*$, and responds $\mathcal{A}$ with the trapdoor $Tr = (T_1, T_2, T_3)$, where
  $$T_1 = h_i^{ar'} = (g^a)^{a_i r'}, T_2 = (g^a)^{r'}, T_3 = g^{r'}.$$
- **Challenge**: $\mathcal{A}$ chooses two keyword sets $\{KW_0^{i*}\}_{i\in[1,I]}$ and $\{KW_1^{i*}\}_{i\in[1,I]}$ with the restriction that no element of $\{PK_R, PK_S, KW_j^{i*}\}_{j\in[0,1],i\in[1,I]}$ has been queried on $\mathcal{O}_C$ or $\mathcal{O}_T$. There are r_I duplicated keywords in the set $\{KW_j^{i*}\}_{j\in[0,1],i\in[1,I]}$. $\mathcal{B}$ retrieves $\{(PK_R, PK_S, KW_j^{i*}), h_j^{i*}, a_j^{i*}, c_j^{i*}\}_{j\in[0,1],i\in[1,I]}$ from H-list and generates the challenge ciphertexts as follows:

1. If, for each $i \in [1, I]$, $c_0^{i*} = c_1^{i*} = 1$, $\mathcal{B}$ aborts the game and outputs a random bit θ' as the guess of θ.
2. Otherwise, there exist a $u \in [1, I]$ such that $c_0^{u*} = 0$ or $c_1^{u*} = 0$. For $i \neq u$, $\mathcal{B}$ generates the challenge ciphertext CT_i^* as Phase 1; for $i = u$, $\mathcal{B}$ picks a bit $\hat{\theta} \in \{0, 1\}$ such that $c_{\hat{\theta}}^* = 0$. Then, it picks $s \in_R \mathbb{Z}_p^*$, sets $r^* = c \cdot s$ and generates the challenge ciphertext $CT_u^* = (C_1^*, C_2^*, C_3^*)$ for $\mathcal{A}$ as:

$$C_1^* = (h_{\hat{\theta}}^*)^{bs} \cdot g^{r^*} = (g^{d/b} \cdot g^{a_{\hat{\theta}}^*})^{bs} \cdot (g^c)^s = (g^{c+d})^s \cdot (g^b)^{a_{\hat{\theta}}^* s} = \mathcal{Z}^s \cdot (g^b)^{a_{\hat{\theta}}^* s}$$
$$C_2^* = (g^a)^{r^*} = (g^{ac})^s, \qquad C_3^* = (g^b)^s.$$

   For $i \neq u$, CT_i^* is random in the view of $\mathcal{A}$. For $i = u$, CT_u^* is a valid ciphertext, if $\mathcal{Z} = g^{c+d}$; otherwise, $\mathcal{Z} = \mathcal{R} \in_R \mathbb{G}_0$ so that C_1^* is a random element in $\mathbb{G}_0$ and CT_u^* is random in the view of $\mathcal{A}$.

- **Phase2**: This phase is the same as Phase 1.
- **Guess**: $\mathcal{A}$ outputs a guess bit θ'' of $\hat{\theta}$. If $\theta'' = \hat{\theta}$, $\mathcal{B}$ guesses $\theta = 0$ which indicates that $\mathcal{Z} = g^{c+d}$ in the above game. Otherwise, $\mathcal{B}$ guesses $\theta = 1$ i.e., $\mathcal{Z} = \mathcal{R}$.

If $\mathcal{Z} = \mathcal{R}$, then CT_u^* is random from the view of $\mathcal{A}$. Hence, $\mathcal{B}$'s probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, g^{ac}, g^{d/b}, \mathcal{Z} = \mathcal{R}\right) = 1\right] = \frac{1}{2}. \tag{8}$$

Else $\mathcal{Z} = g^{c+d}$, then CT_u^* is an available ciphertext and $\mathcal{A}'s$ advantage of guessing θ' is ϵ. Therefore, $\mathcal{B}$'s probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, g^{ac}, g^{d/b}, \mathcal{Z} = g^{c+d}\right) = 0\right] = \frac{1}{2} + \epsilon. \tag{9}$$

Now, we donate the event $\mathcal{B}$ aborts in the above game by $\mathbf{E}_0$ and compute the probability of $\overline{\mathbf{E}_0}$. Note that, $\mathcal{B}$ aborts in the following two cases:

1. $c_i = 0$ in the simulation of $\mathcal{O}_C$ or $\mathcal{O}_T$ in phase 1 and phase 2. We denote this event by $\mathbf{E}_1$ and the probability $\mathbf{E}_1$ doesn't occur is that

$$\Pr\left[\overline{\mathbf{E}_1}\right] = (1 - \delta)^{q_C + q_T - r_C - r_T}.$$

2. For each $i \in [1, I]$, $c_0^{i*} = c_1^{i*} = 1$ challenge phase. We denote this event by $\mathbf{E}_2$ and the probability $\mathbf{E}_2$ doesn't occur is that

$$\Pr\left[\overline{\mathbf{E}_2}\right] = 1 - (1 - \delta)^{2I - r_I}.$$

We sets $\delta = 1 - \sqrt[2I-r_I]{\frac{q_C+q_T-r_C-r_T}{q_C+q_T-r_C-r_T+1}}$, then $\mathcal{B}$ doesn't abort in the above game is that

$$\begin{aligned}
\Pr\left[\overline{\mathbf{E}_0}\right] &= \Pr\left[\overline{\mathbf{E}_1}\right] \cdot \Pr\left[\overline{\mathbf{E}_2}\right] \\
&= \left(\frac{q_C + q_T - r_C - r_T}{q_C + q_T - r_C - r_T + 1}\right)^{\frac{q_C+q_T-r_C-r_T}{2I-r_I}} \cdot \frac{1}{q_C + q_T - r_C - r_T + 1} \\
&\approx \frac{1}{(q_C + q_T - r_C - r_T + 1) \cdot e^{\frac{1}{2I-r_I}}}.
\end{aligned} \tag{10}$$

Hence, $\mathcal{B}$ doesn't abort with a non-negligible probability.

In conclusion, $\mathcal{B}$'s probability to win the above security game is

$$\begin{aligned}\Pr[\theta' = \theta] &= \Pr[\theta' = \theta \wedge \mathbf{E}_0] + \Pr\left[\theta' = \theta \wedge \overline{\mathbf{E}_0}\right] \\ &= \Pr[\theta' = \theta \| \mathbf{E}_0] \cdot \Pr[\mathbf{E}_0] + \Pr\left[\theta' = \theta \| \overline{\mathbf{E}_0}\right] \cdot \Pr\left[\overline{\mathbf{E}_0}\right] \\ &= \frac{1}{2} \cdot \Pr[\mathbf{E}_0] + (\frac{1}{2} + \epsilon) \cdot \Pr\left[\overline{\mathbf{E}_0}\right] \\ &= \frac{1}{2} + \epsilon \cdot \Pr\left[\overline{\mathbf{E}_0}\right].\end{aligned} \tag{11}$$

Theorem 2. *Our scheme is semantically MTP secure against inside keyword guessing attacks in the random oracle model under the DDH assumption.*

Proof. Supposed that a PPT adversary $\mathcal{A}$ can break the MTP security of our scheme with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DDH tuple from a random one with a non-negligible probability. The DDH challenger $\mathcal{C}$ selects $a, b \in_R \mathbb{Z}_p^*$, $\theta \in \{0,1\}$, $\mathcal{R} \in_R \mathbb{G}_0$ at random. Let $\mathcal{Z} = g^{ab}$, if $\theta = 0$, $\mathcal{R}$ else. Next, $\mathcal{C}$ sends $\mathcal{B}$ the DDH tuple $\langle g, g^a, g^b, \mathcal{Z}\rangle$. Then, $\mathcal{B}$ works as follows.

- **Setup**: $\mathcal{B}$ selects a collusion resistant hash function H and sets $PK_R = g^a, PK_S = g^b$ which implies $SK_R = a, SK_S = b$. Then, it sends $PP = (e, \mathbb{G}_0, \mathbb{G}_T, p, g, H)$ and PK_R, PK_S to $\mathcal{A}$.
- **Phase1**: This phase is the same as that of Theorem 1, except that $\mathcal{B}$ answers the hash queries in a different way.

  Hash Oracle $\mathcal{O}_H$: $\mathcal{B}$ maintains the H-list $\{(PK_R, PK_S, KW_j), h_j, a_j, c_j\}$ and responds the hash query (PK_R, PK_S, KW_i) as follows:
  1. If (PK_R, PK_S, KW_i) has already existed in the H-list, $\mathcal{B}$ responds $\mathcal{A}$ with $H(PK_R, PK_S, KW_i) = h_i \in \mathbb{G}_0$.
  2. Otherwise, $\mathcal{B}$ flips a random coin $c_i \in \{0,1\}$ with the probability $\Pr[c_i = 0] = \delta$.
  3. $\mathcal{B}$ picks a random $a_i \in_R \mathbb{Z}_p^*$. If $c_i = 0$, it sets
  $$H(PK_R, PK_S, KW_i) = h_i = g^{b \cdot a_i} \in \mathbb{G}_0;$$
  If $c_i = 1$, it sets
  $$H(PK_R, PK_S, KW_i) = h_i = g^{a_i} \in \mathbb{G}_0$$
  .
  4. $\mathcal{B}$ adds the tuple $\{(PK_R, PK_S, KW_j), h_j, a_j, c_j\}$ into the H-list and responds $\mathcal{A}$ with $H(PK_R, PK_S, KW_i) = h_i$.

  Ciphertext Oracle $\mathcal{O}_C$: $\mathcal{B}$ answers the ciphertext query in the same way as in the proof of Theorem 1.

  Trapdoor Oracle $\mathcal{O}_T$: $\mathcal{B}$ answers the trapdoor query in the same way as in the proof of Theorem 1.

- **Challenge**: $\mathcal{A}$ chooses two keyword sets $\{KW_0^{i*}\}_{i\in[1,I]}$ and $\{KW_1^{i*}\}_{i\in[1,I]}$ with the restriction that no element of $\{PK_R, PK_S, KW_j^{i*}\}_{j\in[0,1],i\in[1,I]}$ has been queried on $\mathcal{O}_C$ or $\mathcal{O}_T$. There are r_I duplicated keywords in the set $\{KW_j^{i*}\}_{j\in[0,1],i\in[1,I]}$. $\mathcal{B}$ retrieves $\{(PK_R, PK_S, KW_j^{i*}), h_j^{i*}, a_j^{i*}, c_j^{i*}\}_{j\in[0,1],i\in[1,I]}$ from H-list and generates the challenge trapdoors as follows:
  1. If, for each $i \in [1, I]$, $c_0^{i*} = c_1^{i*} = 1$, $\mathcal{B}$ aborts the game and outputs a random bit θ' as the guess of θ.
  2. Otherwise, there exist a $u \in [1, I]$ such that $c_0^{u*} = 0$ or $c_1^{u*} = 0$. For $i \neq u$, $\mathcal{B}$ generates the challenge trapdoor Tr_i^* as Phase 1; for $i = u$, $\mathcal{B}$ picks a bit $\hat{\theta} \in \{0, 1\}$ such that $c_{\hat{\theta}}^* = 0$. Then, it picks $r' \in_R \mathbb{Z}_p^*$, and generates the challenge trapdoor $Tr_u^* = (T_1^*, T_2^*, T_3^*)$ for $\mathcal{A}$ as:

$$T_1^* = (h_{\hat{\theta}}^*)^{ar'} = (g^{ba_{\hat{\theta}}^*})^{ar'} = (g^{ab})^{a_{\hat{\theta}}^* r'} = \mathcal{Z}^{a_{\hat{\theta}}^* r'}$$
$$T_2^* = (g^a)^{r'}, \qquad T_3^* = g^{r'}.$$

  For $i \neq u$, Tr_i^* is random in the view of $\mathcal{A}$. For $i = u$, Tr_u^* is a valid ciphertext, if $\mathcal{Z} = g^{ab}$; otherwise, $\mathcal{Z} = \mathcal{R} \in_R \mathbb{G}_0$ so that T_1^* is a random element in $\mathbb{G}_0$ and Tr_u^* is random in the view of $\mathcal{A}$.
- **Phase2**: This phase is the same as Phase 1.
- **Guess**: $\mathcal{A}$ outputs a guess bit θ'' of $\hat{\theta}$. If $\theta'' = \hat{\theta}$, $\mathcal{B}$ guesses $\theta = 0$ which indicates that $\mathcal{Z} = g^{ab}$ in the above game. Otherwise, $\mathcal{B}$ guesses $\theta = 1$ i.e., $\mathcal{Z} = \mathcal{R}$.

If $\mathcal{Z} = \mathcal{R}$, then CT_u^* is random from the view of $\mathcal{A}$. Hence, $\mathcal{B}$'s probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, \mathcal{Z} = \mathcal{R}\right) = 1\right] = \frac{1}{2}. \tag{12}$$

Else $\mathcal{Z} = g^{ab}$, then Tr_u^* is an available ciphertext and $\mathcal{A}'s$ advantage of guessing θ' is ϵ. Therefore, $\mathcal{B}$'s probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, \mathcal{Z} = g^{ab}\right) = 0\right] = \frac{1}{2} + \epsilon. \tag{13}$$

Similarly with Theorem 1, $\mathcal{B}$ doesn't abort in the above game with a non-negligible probability $\Pr\left[\overline{\mathbf{E}_0}'\right]$. So that, $\mathcal{B}$'s probability to win the above security game is $\frac{1}{2} + \epsilon \cdot \Pr\left[\overline{\mathbf{E}_0'}\right]$.

5 PAEKS with simplified key management

In our first scheme, both sender and receiver generate their public and secret keys by their own. To guarantee the integrity and validity of these keys, we apply a key generation (KGC) center to authorize a sender associated with an unique identity ID_S or a receiver with ID_R.

5.1 Construction

- **Setup**$(1^\lambda, \mathcal{L}) \rightarrow (PP)$: Given a security parameter λ, the algorithm chooses a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_0$ and $\mathbb{G}_T$ are groups with prime order p and g is the generator of $\mathbb{G}_0$, and two hash functions $H_1 : \{0,1\}^* \longrightarrow \mathbb{G}_0$ and $H_2 : \mathcal{ID} \longrightarrow \mathbb{G}_0$. It picks a random $\alpha \in_R \mathbb{Z}_p^*$, then outputs the public parameter $PP = (e, \mathbb{G}_0, \mathbb{G}_T, p, g, H_1, H_2)$ and the master secret key $MSK = \alpha$.
- $\mathbf{KeyGen_R}(PP, MSK) \rightarrow (PK_R, SK_R)$: The KGC generates a secret key $H_2(ID_R)^\alpha$ for the receiver. The receiver picks a random $x \in_R \mathbb{Z}_p^*$, then sets $PK_R = (g^x, ID_R)$ and $SK_R = (x, H_2(ID_R)^\alpha)$.
- $\mathbf{KeyGen_S}(PP) \rightarrow (PK_S, SK_S)$: The KGC generates $H_2(ID_S)^\alpha$ for the sender. The sender picks a random $y \in_R \mathbb{Z}_p^*$, then sets $PK_S = (g^y, ID_S)$ and $SK_S = (y, H_2(ID_S)^\alpha)$.
- **PAEKS**$(PP, PK_R, PK_S, SK_S, KW) \rightarrow CT$: It randomly chooses $s, r \in_R \mathbb{Z}_p^*$, then computes $K = e(H_2(ID_S)^\alpha, H_2(ID_R))$ and
$$C_1 = H_1(ID_R, ID_S, KW, K)^{ys} g^r, C_2 = g^{xr}, C_3 = g^{ys}. \tag{14}$$
Finally, it outputs $CT = (C_1, C_2, C_3)$.
- **TrapGen**$(PK_R, PK_S, SK_R, KW') \rightarrow Tr$: It randomly chooses $r' \in_R \mathbb{Z}_p^*$, then computes $K' = e(H_2(ID_S), H_2(ID_R)^\alpha)$ and
$$T_1 = H_1(ID_R, ID_S, KW', K')^{xr'}, T_2 = g^{xr'}, T_3 = g^{r'}. \tag{15}$$
Finally, it outputs $Tr = (T_1, T_2, T_3)$.
- **Test**(PP, Ct, Tr): The algorithm checks whether the following equation holds or not:
$$e(C_1, T_2) = e(T_1, C_3) \cdot e(T_3, C_2). \tag{16}$$
If so, it outputs 1; otherwise it outputs 0.

5.2 Correctness

The receiver's key pair is $PK_R = (g^x, ID_R); SK_R = (x, H_2(ID_R)^\alpha)$ and the sender's key pair is $PK_S = (g^y, ID_S); SK_S = (y, H_2(ID_S)^\alpha)$, while H is a collusion resistant hash function. If the ciphertext keyword KW is identical to the target keyword KW', we have the following equations:

$$K = e(H_2(ID_S)^\alpha, H_2(ID_R)) = e(H_2(ID_S), H_2(ID_R)^\alpha) = K' \tag{17}$$

$$\begin{aligned} e(C_1, T_2) =& e(H(ID_R, ID_S, KW, K)^{ys} g^r, g^{xr'}) \\ =& e(H(ID_R, ID_S, KW, K), g)^{xysr'} \cdot e(g, g)^{xrr'} \\ =& e(H(ID_R, ID_S, KW', K')^{xr'}, g^{ys}) \cdot e(g^{r'}, g^{xr}) \\ =& e(T_1, C_3) \cdot e(T_3, C_2). \end{aligned} \tag{18}$$

The above equation holds with probability 1, if $KW = KW'$ and $K = K'$; it doesn't hold with overwhelming probability when $KW \neq KW'$ or $K \neq K'$.

5.3 Security proof

Theorem 3. *Our second scheme is MCI secure against outside chosen keyword attacks in the random oracle model under the DBDH assumption.*

Proof. Supposed that a PPT adversary $\mathcal{A}$ can break the MCI security of our second scheme with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random one with a non-negligible probability. The DBDH challenger $\mathcal{C}$ selects $a, b, c \in_R \mathbb{Z}_p^*$, $\theta \in \{0,1\}$, $\mathcal{R} \in_R \mathbb{G}_0$ at random. Let $\mathcal{Z} = e(g,g)^{abc}$, if $\theta = 0$, $\mathcal{R}$ else. Next, $\mathcal{C}$ sends $\mathcal{B}$ the DBDH tuple $\langle g, g^a, g^b, g^c, \mathcal{Z}\rangle$. Then, $\mathcal{B}$ plays the role of simulator in the following security game.

- **Setup**: $\mathcal{B}$ selects two hash function H_1, H_2. H_1 is a collusion resistant and H_2 serves as a random oracle.
- **Phase1**: The adversary $\mathcal{A}$ may issue at most q_{H_2} queries to the hash oracle $\mathcal{O}_{H_2}$, secret key oracle $\mathcal{O}_{SK}$, the ciphertext oracle $\mathcal{O}_C$ and the trapdoor oracle $\mathcal{O}_T$. We make some necessary restriction: q_{H_2} are polynomial; $\mathcal{A}$ isn't allowed to repeat a query to the hash oracle $\mathcal{O}_{H_2}$ or secret key oracle $\mathcal{O}_{SK}$, but may repeat queries to $\mathcal{O}_C$ and $\mathcal{O}_T$; $\mathcal{A}$ cannot issue any query $\mathcal{O}_{SK}$, $\mathcal{O}_C$ or $\mathcal{O}_T$ before issuing the associated ID to $\mathcal{O}_{H_2}$.

  $\mathcal{B}$ simulates these oracles as follows:

  Hash Oracle $\mathcal{O}_{H_2}$: $\mathcal{B}$ maintains a list of tuple $\{ID_j, H_2(ID_j), v_j, g^{z_j}, z_j\}$ as the "H_2-list", which is initially empty. Given a hash query ID_i, $\mathcal{B}$ responds as follows:

  1. If ID_i has already existed in the H_2-list, $\mathcal{B}$ responds $\mathcal{A}$ with $H_2(ID_i)$ and the public key PK_{ID_i} associated with ID_i.
  2. Otherwise, if $i = i^*$, i.e., $ID_i = ID_R^*$, $\mathcal{B}$ picks $x \in_R \mathbb{Z}_p^*$, sets $H_2(ID_R^*) = g^a$, adds $\{ID_R^*, g^a, v_R = \perp, g^x, x\}$ to H_2-list, and then responds $\mathcal{A}$ with $H_2(ID_R^*)$ and $PK_{ID_R^*} = (g^x, ID_R^*)$; if $i = j^*$, i.e., $ID_i = ID_S^*$, $\mathcal{B}$ picks $y \in_R \mathbb{Z}_p^*$, sets $H_2(ID_S^*) = g^b$, adds $\{ID_S^*, g^b, v_R = \perp, g^y, y\}$ to H_2-list, and then responds $\mathcal{A}$ with $H_2(ID_S^*)$ and $PK_{ID_S^*} = (g^y, ID_S^*)$; else, $\mathcal{B}$ picks a random $z_i, v_i \in_R \mathbb{Z}_p^*$, sets $H_2(ID_i) = g^{v_i}$, adds $\{ID_i, g^{v_i}, v_i, g^{z_i}, z_i\}$ to H_2-list, and then responds $\mathcal{A}$ with $H_2(ID_i)$ and $PK_{ID_i} = (g^{z_i}, ID_i)$.

  Secret Key Oracle $\mathcal{O}_{SK}$: Given a secret key query ID_i, if $ID_i = ID_S^*$ or $ID_i = ID_R^*$, $\mathcal{B}$ aborts the game and outputs a random bit θ' as the guess of θ. Otherwise, it retrieves the tuple $\{ID_i, g^{v_i}, v_i, g^{z_i}, z_i\}$ from H_2-list and returns $SK_{ID_i} = \{z_i, (g^c)^{v_i}\}$.

  Ciphertext Oracle $\mathcal{O}_C$: Given a ciphertext query $(PK_{ID_i}, PK_{ID_j}, KW)$, $\mathcal{B}$ picks $s, r \in_R \mathbb{Z}_p^*$, and responds $\mathcal{A}$ with the ciphertext $Ct = (C_1, C_2, C_3)$ according to the following two cases:

  1. $(ID_i, ID_j) = (ID_R^*, ID_S^*)$ or $(ID_i, ID_j) = (PK_{ID_S^*}, PK_{ID_R^*})$. Without loss of generality, we assume $ID_i = ID_R^*$. $\mathcal{B}$ sets $K = \mathcal{Z}$ and sets

$$C_1 = H_1(ID_R^*, ID_S^*, KW, \mathcal{Z})^{ys} g^r, C_2 = g^{xr}, C_3 = g^{ys}.$$

2. $ID_i \notin \{ID_R^*, ID_S^*\}$ or $ID_j \notin \{ID_R^*, ID_S^*\}$. Without loss of generality, we assume $ID_i \notin \{ID_R^*, ID_S^*\}$. $\mathcal{B}$ computes

$$K = e(H_2(ID_i), H_2(ID_j)^c) = e(g^c, H_2(ID_j))^{z_i}$$

and sets

$$C_1 = H_1(ID_i, ID_j, KW, K)^{ys} g^r, C_2 = g^{xr}, C_3 = g^{ys}.$$

Trapdoor Oracle $\mathcal{O}_T$: Given a trapdoor query $(PK_{ID_i}, PK_{ID_j}, KW')$, $\mathcal{B}$ picks $r' \in_R \mathbb{Z}_p^*$, and responds $\mathcal{A}$ with the trapdoor $Tr = (T_1, T_2, T_3)$ according to the following two cases:

1. $(ID_i, ID_j) = (ID_R^*, ID_S^*)$ or $(ID_i, ID_j) = (PK_{ID_S^*}, PK_{ID_R^*})$. Without loss of generality, we assume $ID_i = ID_R^*$. $\mathcal{B}$ sets $K' = \mathcal{Z}$ and sets

$$T_1 = H_1(ID_R^*, ID_S^*, KW', \mathcal{Z})^{xr'}, T_2 = g^{xr'}, T_3 = g^{r'}.$$

2. $ID_i \notin \{ID_R^*, ID_S^*\}$ or $ID_j \notin \{ID_R^*, ID_S^*\}$. Without loss of generality, we assume $ID_i \notin \{ID_R^*, ID_S^*\}$. $\mathcal{B}$ computes

$$K' = e(H_2(ID_i), H_2(ID_j)^c) = e(g^c, H_2(ID_j))^{z_i}$$

and sets

$$T_1 = H_1(ID_i, ID_j, KW', K')^{xr'}, T_2 = g^{xr'}, T_3 = g^{r'}.$$

- **Challenge**: $\mathcal{A}$ chooses two keyword sets $\{KW_0^{i*}\}_{i\in[1,I]}$ and $\{KW_1^{i*}\}_{i\in[1,I]}$ that no element of $\{PK_{ID_R^*}, PK_{ID_S^*}, KW_j^{i*}\}_{j\in[0,1],i\in[1,I]}$ has been queried on $\mathcal{O}_C$ or $\mathcal{O}_T$. For each $j \in \{0,1\}, i \in [1, I]$, $\mathcal{B}$ picks $s_j^{i^*}, r_j^{i^*} \in_R \mathbb{Z}_p^*$, and generates each challenge ciphertext $CT_j^{i^*} = (C_{1,j}^{i*}, C_{2,j}^{i*}, C_{3,j}^{i*})$ as:

$$C_{1,j}^{i*} = H_1(ID_R^*, ID_S^*, KW_j^{i*}, \mathcal{Z})^{ys_j^{i^*}} g^{r_j^{i^*}}, C_{2,j}^{i*} = g^{xr_j^{i^*}}, C_{3,j}^{i*} = g^{ys_j^{i^*}}.$$

- **Phase2**: This phase is the same as Phase 1.
- **Guess**: $\mathcal{A}$ outputs a guess bit θ'' of $\hat{\theta}$. If $\theta'' = \hat{\theta}$, $\mathcal{B}$ guesses $\theta = 0$ which indicates that $\mathcal{Z} = e(g,g)^{abc}$ in the above game. Otherwise, $\mathcal{B}$ guesses $\theta = 1$ i.e., $\mathcal{Z} = \mathcal{R}$.

If $\mathcal{Z} = \mathcal{R}$, then $\{CT_j^{i^*}\}_{j\in\{0,1\},i\in[1,I]}$ is random from the view of $\mathcal{A}$. Hence, $\mathcal{B}$'s probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, g^{ac}, g^{d/b}, \mathcal{Z} = \mathcal{R}\right) = 1\right] = \frac{1}{2}. \tag{19}$$

Else $\mathcal{Z} = e(g,g)^{abc}$, then $\{CT_j^{i^*}\}_{j\in\{0,1\},i\in[1,I]}$ is a set of available ciphertexts and $\mathcal{A}'s$ advantage of guessing θ' is ϵ. Therefore, $\mathcal{B}$'s probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^a, g^b, g^{ac}, g^{d/b}, \mathcal{Z} = g^{c+d}\right) = 0\right] = \frac{1}{2} + \epsilon. \tag{20}$$

Since, $\mathcal{B}$ doesn't abort in the security game with the probability $\frac{1}{q_{H_2}(q_{H_2}-1)}$, its advantage to win the above security game is $\frac{\epsilon}{2q_{H_2}(q_{H_2}-1)}$.

Theorem 4. *Our second scheme is MTP secure against inside keyword guessing attacks in the random oracle model under the DBDH assumption.*

Proof. The proof is similar with that of Theorem 3 except that $\mathcal{B}$ picks generates the challenge trapdoor $Tr_j^{i^*} = (T_{1,j}^{i^*}, T_{2,j}^{i^*}, T_{3,j}^{i^*})$, for each $j \in \{0,1\}, i \in [1, I]$, as:

$$T_{1,j}^{i^*} = H_1(ID_R^*, ID_S^*, KW_j^{i*}, \mathcal{Z})^{x\hat{r}_j^{i^*}}, T_{2,j}^{i*} = g^{x\hat{r}_j^{i^*}}, T_{3,j}^{i*} = g^{\hat{r}_j^{i^*}},$$

where $\hat{r}_j^{i^*} \in_R \mathbb{Z}_p^*$ chosen by $\mathcal{B}$. For simplicity, we omit the detailed proof here.

6 Conclusion

In this paper, we initially propose two public-key authenticated encryption with keyword search schemes achieving both MCI security and MTP security simultaneously. The trapdoor generation algorithms in our schemes are all probabilistic. In the area of PAEKS, combining our schemes and [10], it would be able to design a secure designated PAEKS scheme with both MCI security and MTP security. For the compact of this paper, we omit it here.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In: Shoup, V. (ed.) Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings. Lecture Notes in Computer Science, vol. 3621, pp. 205–222. Springer (2005)
2. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: TCC 2007. pp. 535–554 (2007)
3. Byun, J.W., Rhee, H.S., Park, H., Lee, D.H.: Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: SDM 2006. pp. 75–83 (2006)
4. Chen, R., Mu, Y., Yang, G., Guo, F., Wang, X.: Dual-server public-key encryption with keyword search for secure cloud storage. IEEE Trans. Inf. Forensics Secur. **11**(4), 789–798 (2016)
5. Fei MENG, Leixiao CHENG, M.W.: Abdks: Attribute-based encryption with dynamic keyword search in fog computing. Frontiers of Computer Science
6. Fuhr, T., Paillier, P.: Decryptable searchable encryption. In: Susilo, W., Liu, J.K., Mu, Y. (eds.) Provable Security, First International Conference, ProvSec 2007, Wollongong, Australia, November 1-2, 2007, Proceedings. Lecture Notes in Computer Science, vol. 4784, pp. 228–236. Springer (2007)
7. Golle, P., Staddon, J., Waters, B.R.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings. Lecture Notes in Computer Science, vol. 3089, pp. 31–45. Springer (2004)

8. He, D., Ma, M., Zeadally, S., Kumar, N., Liang, K.: Certificateless public key authenticated encryption with keyword search for industrial internet of things. IEEE Trans. Ind. Informatics **14**(8), 3618–3627 (2018)
9. Huang, Q., Li, H.: An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Inf. Sci. **403**, 1–14 (2017)
10. Li, H., Huang, Q., Shen, J., Yang, G., Susilo, W.: Designated-server identity-based authenticated encryption with keyword search for encrypted emails. Inf. Sci. **481**, 330–343 (2019)
11. Noroozi, M., Eslami, Z.: Public key authenticated encryption with keyword search: revisited. IET Inf. Secur. **13**(4), 336–342 (2019)
12. Park, D.J., Kim, K., Lee, P.J.: Public key encryption with conjunctive field keyword search. In: WISA 2004. pp. 73–86 (2004)
13. Qin, B., Chen, Y., Huang, Q., Liu, X., Zheng, D.: Public-key authenticated encryption with keyword search revisited: Security model and constructions. Inf. Sci. **516**, 515–528 (2020)
14. Rhee, H.S., Lee, D.H.: Keyword updatable PEKS. In: Kim, H., Choi, D. (eds.) Information Security Applications - 16th International Workshop, WISA 2015, Jeju Island, Korea, August 20-22, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9503, pp. 96–109. Springer (2015)
15. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. **83**(5), 763–771 (2010)
16. Shao, J., Cao, Z., Liang, X., Lin, H.: Proxy re-encryption with keyword search. Inf. Sci. **180**(13), 2576–2587 (2010)
17. Shi, E., Bethencourt, J., Chan, T.H., Song, D.X., Perrig, A.: Multi-dimensional range query over encrypted data. In: S&P 2007. pp. 350–364 (2007)
18. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000. pp. 44–55. IEEE Computer Society (2000)
19. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: INFOCOM 2014. pp. 522–530 (2014)