Hindawi
Wireless Communications and Mobile Computing
Volume 2020, Article ID 8816172, 11 pages
https://doi.org/10.1155/2020/8816172

Research Article

An Efficient Searchable Public-Key Authenticated Encryption for Cloud-Assisted Medical Internet of Things

Tianyu Chi,[1] Baodong Qin,[1,2] and Dong Zheng[1,3]

[1]*School of Cyberspace Security, Xi'an University of Posts & Telecommunications, Xi'an, Shaanxi, China*
[2]*State Key Laboratory of Integrated Services Networks, Xidian University, Xi'an, China*
[3]*Westone Cryptologic Research Center, Beijing, China*

Correspondence should be addressed to Baodong Qin; qinbaodong@foxmail.com

Received 27 March 2020; Revised 13 June 2020; Accepted 17 June 2020; Published 14 July 2020

Academic Editor: Huaqun Wang

In recent years, it has become popular to upload patients' medical data to a third-party cloud server (TCS) for storage through medical Internet of things. It can reduce the local maintenance burden of the medical data and importantly improve accuracy in the medical treatment. As remote TCS cannot be fully trusted, medical data should be encrypted before uploading, to protect patients' privacy. However, encryption makes search capabilities difficult for patients and doctors. To address this issue, Huang et al. recently put forward the notion of Public-key Authenticated Encryption with Keyword Search (PAEKS) against inside keyword guessing attacks. However, the existing PAEKS schemes rely on time-consuming computation of parings. Moreover, some PAEKS schemes still have security issues in a multiuser setting. In this paper, we propose a new and efficient PAEKS scheme, which uses the idea of Diffie-Hellman key agreement to generate a shared secret key between each sender and receiver. The shared key will be used to encrypt keywords by the sender and to generate search trapdoors by the receiver. We prove that our scheme is semantically secure against inside keyword guessing attacks in a multiuser setting, under the oracle Diffie-Hellman assumption. Experimental results demonstrate that our PAEKS scheme is more efficient than that of previous ones, especially in terms of keyword searching time.

1. Introduction

In today's society, almost all medical service providers will use some form of electronic medical record system [1]. Specifically, medical Internet of things (MIoT) has become a new technology to gather data from patients by small wearable devices or implantable sensors. With the increasing number of medical data, the burden of hospital storage equipment is heavy, and it needs a professional person to maintain. If the hardware storage device is damaged and data is lost due to other force majeure factors, it will lead to very serious consequences. The most important way to solve this problem is to upload the data to the third-party cloud server (TCS). However, after the data is uploaded to the TCS, the patient's privacy will not be guaranteed. Once the cloud server managers or external malicious attackers steal the data, it will cause data leakage and other problems [2].

In order to solve the problem of data security, the best way is to encrypt the data and then upload the result to TCS. But when medical service providers want to retrieve the electronic medical records of patients, it becomes more difficult. First, doctors need to download all encrypted data to a local server and then decrypt it locally. After that, they can search for the desired results in the plaintext medical data. However, this process is very cumbersome and impractical for most applications. Due to the powerful cloud computing, medical institutions hope that the cloud server can complete the retrieval function instead of doing it themselves. But if the key is sent to the cloud server, the patient's private data still has the risk of exposure.

To address the above security issues, the conception of (symmetric-key) searchable encryption (SE) was proposed by Song et al. [3]. It is a powerful technology that allows the cloud server to search on encrypted data using some search

trapdoors generated by the local data users. In 2004, Boneh et al. [4] proposed a public-key version of SE, namely, Public-key Encryption with Keyword Search (PEKS). This scheme embeds keywords in public-key encryption and is very suitable for scenarios of a multiuser data sharing setting, e.g., medical data sharing. There are three parties in the PEKS scheme: cloud server, data sender, and data receiver. The sender (e.g., patient) has a lot of privacy files F_i and wants to share them with the receiver (e.g., doctor). First, the sender extracts the keyword w_i from each file F_i, encrypts the keyword with the PEKS scheme, and then encrypts each file with other encryption schemes (not necessarily the same as the PEKS scheme). Let the keyword cipher text be C_{w_i}. The sender uploads all cipher texts to the TCS. In order to search whether there is a document containing the keyword w in the encrypted document, the receiver generates a search trapdoor T_w of the keyword w and sends the trapdoor to the cloud server. After the server receives T_w, it checks whether each keyword cipher text matches with the search trapdoor. If so, it indicates that the corresponding encrypted document must contain the desired keyword. After that, the results are returned to the receiver, and the receiver can get the required plaintext data by decrypting the encrypted documents.

As mentioned in Figure 1, we will apply searchable encryption to telemedicine services, where the patient is the sender and the medical service provider is the receiver. Each patient can encrypt and upload their own electronic medical record to the cloud server. When the patient wants to see a doctor remotely, the doctor can retrieve the medical data information related to some disease on the third-party cloud server according to the keyword information of the patient. In this process, doctors will only get data related to a certain disease and will not expose other information (such as name) of patients.

However, PEKS inherently has a disadvantage to resist against keyword guessing attacks (KGA). Ideally, a keyword space can be considered infinite. In practice, however, this is not the case. In real life, users often use a limited number of keywords because of their living habits, which leads to the transformation of the original polynomial space into an affixed and low-entropy space. In this case, the adversary can guess the keywords contained in the searching trapdoor as follows: First, the adversary guesses all the keyword spaces of the user and then generates keyword cipher text one by one. The adversary checks the trapdoor requested by the user one by one with keyword cipher texts generated by itself. If there is coincidentally the same situation, the adversary can obtain the keyword information retrieved by the user, thus exposing the privacy of the user. This kind of attack can be easily mounted by the cloud server, as the cloud server has users' searching trapdoors. Such attack is often called inside keyword guessing attacks (IKGA).

To resist against KGA is very challenging. Recently, many methods [5–11] were proposed to prevent KGA on PEKS schemes; however, most of them were later proven insecure [12–15]. In 2017, Huang and Li [16] proposed a new primitive, namely, Public-key Authenticated Encryption with Keyword Search (PAEKS), to solve the problem of inside KGA. In PAEKS, the data sender not only encrypts a keyword but also authenticates it, so that a search trapdoor can only match with the corresponding data sender. PAEKS is also applicable to cloud-assisted MIoT, as in general, the doctor just searches on a designated patient's medical data. However, the proposed concrete PAEKS scheme still has some security issues [17–19]. In particular, Noroozi and Eslami [18] pointed out that it cannot handle multiuser settings and provided an improvement security model for PAEKS in a multiuser setting.

1.1. Our Contribution. In this paper, we research on new and efficient construction of PAEKS schemes in a multiuser setting for cloud-assisted MIoT. Our main contributions are as follows:

(i) We observe that in PAEKS, both the data sender and data receiver hold a pair of public/secret keys. If they can compute a shared key without any interaction, then the shared key can be viewed as a secret key of a symmetric searchable encryption scheme. Inspired by this, we propose an efficient PAEKS scheme, which involves the (noninteractive) Diffie-Hellman key exchange scheme to compute the shared key and Song et al.'s SSE scheme to encrypt keywords. It removes the usage of time-consuming operation of pairing in previous PAEKS schemes

(ii) We show that our scheme is semantically secure against IKGA in a multiuser setting under the oracle Diffie-Hellman assumption [20]. Specially, it satisfies both cipher text indistinguishability and trapdoor indistinguishability

(iii) We compare our scheme with some related PAEKS scheme in terms of security and computation efficiency and also do some experiments to demonstrate the efficiency of our schemes for protecting the privacy of cloud-assisted MIoT data. Experiment results show that our scheme is more efficient than that of previous ones, especially in terms of keyword searching time

1.2. Paper Organization. In the next section, we will briefly introduce some cryptographic primitives. Our main construction of the PAEKS scheme and its security proof are given in Section 3. In Section 4, we compare the efficiency of our scheme with that of other related PAEKS schemes. Finally, we summarize the paper in Section 5.

2. Preliminaries

In this section, we recall some basic conceptions of cryptographic primitives that will be used in this paper, including cyclic group, hardness assumption, pseudorandom functions, syntax of PAEKS, and its security model.

2.1. Cyclic Group. Let G be a group with order p. We say that G is a cyclic group, if the group G can be generated by a single element $g \in G$. That is, every element $h \in G$ has the form $h = g^x$ for some exponent $x \in \mathbb{Z}_p$. We call g to be a

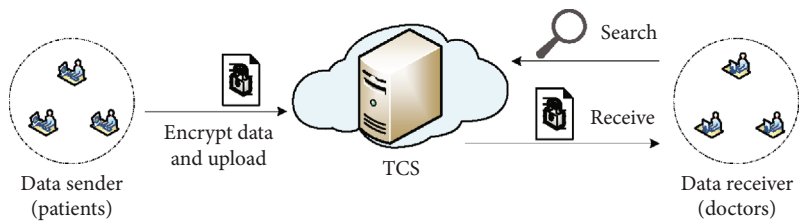

Figure 1: Telemedicine service based on searchable encryption.

generator of the group. In our scheme, we use a cyclic group with a prime order; i.e., p is a prime. In this case, any group element except the identity will be a generator.

2.2. Oracle Diffie-Hellman (ODH) Problem [20]. Let G be a cyclic group with prime order p and a generator g. Let H be a hash function from G to some n-bit length space $\{0,1\}^n$. The ODH problem states that given a tuple (g, g^x, g^y, T) and an oracle $\mathcal{O}_{g^x}(\cdot)$, to decide whether T is $H(g^{xy})$ or a random string from $\{0,1\}^n$, here, x and y are randomly chosen from $\mathbb{Z}_p$, and the oracle returns $H(h^x)$ for each $h \in G$. Let $\mathcal{A}$ be any probabilistic polynomial time (PPT) algorithm. We say that $\mathcal{A}$ breaks the ODH problem over group G and H with advantage at most ϵ_{odh}, if

$$\left|\Pr\left[\mathcal{A}^{\mathcal{O}_{g^x}(\cdot)}(g, g^x, g^y, H(g^{xy})) = 1\right] - \Pr\left[\mathcal{A}^{\mathcal{O}_{g^x}(\cdot)}(g, g^x, g^y, T) = 1\right]\right| < \epsilon_{\text{odh}}. \quad (1)$$

Definition 1 (ODH assumption). We say that the ODH assumption holds over group G and H, if for any PPT algorithm $\mathcal{A}$, its advantage ϵ_{odh} in solving the ODH problem is negligible in κ (the bit length of p).

2.3. Pseudorandom Functions (PRFs). A pseudorandom function is a family of functions such that for a random choice from the family, its input/output behavior is computationally indistinguishable from that of a random function. A formal definition of PRFs is given below.

Definition 2 (PRFs). Let $f : \mathcal{K} \times \mathcal{X} \longrightarrow \mathcal{Y}$ be a family of functions indexed with key space $\mathcal{K}$ from $\mathcal{X}$ to $\mathcal{Y}$. We say that f is an $\epsilon_f - \text{securePRF}_s$ if

(1) Given a key $k \in \mathcal{K}$ and an input $x \in \mathcal{X}$, there is an efficient algorithm to compute the output $y = f_k(x)$

(2) For any PPT algorithm $\mathcal{A}$ that makes at most polynomial number of oracle queries, the following advantage is at most ϵ_f:

where $F = \{f : \mathcal{X} \longrightarrow \mathcal{Y}\}$ and the oracles are given an input $x \in \mathcal{X}$ and output the corresponding image of the function.

$$\left|\Pr\left[\mathcal{A}\left(\mathcal{O}_{g^x}(\cdot)\right) = 1 : k \longleftarrow \mathcal{K}\right] - \Pr\left[\mathcal{A}\left(\mathcal{O}_{g^x}(\cdot)\right) = 1 : f \longleftarrow F\right]\right| \le \epsilon_f, \quad (2)$$

The above definition indicates that, given any polynomial number of valid input/output pairs $(x_i, f_k(x_i))$, no PPT adversary can predicate $f_k(x)$ for a new and distinct input x. Specifically, $f_k(x)$ is computationally indistinguishable from a random $y \in \mathcal{Y}$.

2.4. PAEKS and Security Model. The notion of Public-key Authenticated Encryption with Keyword Search (PAEKS) was first proposed in [16] to protect the privacy of a keyword against inside keyword guessing attacks. It involves the public/secret key pair into the cipher text to prevent keyword guessing attacks by the insider server. We first recall its definition.

Definition 3 (syntax of PAEKS). A PAEKS scheme consists of the following six PPT algorithms:

(i) *Setup* (λ). This is the global parameter generation algorithm. It takes the security parameter λ as input and outputs global system parameter Param

(ii) $KeyGen_S$ *(Param*). This is the sender's key generation algorithm. It takes the global system parameter Param as input and outputs a public/secret key pair $(\text{pk}_\text{S}, \text{sk}_\text{S})$

(iii) $KeyGen_R$ *(Param*). This is the receiver's key generation algorithm. It takes the global system parameter Param as input and outputs a public/secret key pair $(\text{pk}_\text{R}, \text{sk}_\text{R})$

(iv) *PAEKS* (sk_S, pk_R, w). This is the keyword encryption algorithm performed by the sender. It takes the sender's secret key sk_S, the receiver's public key pk_R, and a keyword w as input and outputs a PAEKS cipher text C of the keyword w

(v) *Trapdoor* (sk_R, pk_S, w). This is the trapdoor generation algorithm performed by the receiver. It takes the receiver's secret key sk_R, the sender's public key pk_S, and a keyword w as input and outputs a trapdoor T_w

(vi) *Test* (T_w, C, pk_S, pk_R). This is the test algorithm performed by the cloud server. It takes a trapdoor T_w, a PAEKS cipher text C, the sender's public key pk_S, and the receiver's public key pk_R as input and outputs 1 if C and T_w contain the same keyword and 0 otherwise

Next, we recall the improved security model for PAEKS in a multiuser setting by Noroozi and Eslami [18]. It includes trapdoor indistinguishability (TI) and cipher text indistinguishability (CI). Both of them are described through games played between an adversary $\mathscr{A}$ and the challenger $\mathscr{C}$.

Definition 4 (TI security game). The TI security game is described as follows:

(i) *Initialization*. Given a security parameter λ, the challenger generates the global system parameter. Then, the challenger generates the receiver's public/secret keys $(\mathrm{pk}_R, \mathrm{sk}_R)$ and the sender's public/secret keys $(\mathrm{pk}_S, \mathrm{sk}_S)$. It executes the adversary $\mathscr{A}$ on input $(\mathrm{Param}, \mathrm{pk}_S, \mathrm{pk}_R)$

(ii) *Phase 1*. The adversary A is permitted to adaptively query the following two oracles polynomial times:

(a) *Cipher Text Oracle* $\mathscr{O}_C(w, pk)$. Given a keyword w and a public key pk, the challenger computes the cipher text C by running the algorithm PAEKS $(\mathrm{sk}_S, \mathrm{pk}, w)$ and returns the cipher text to $\mathscr{A}$

(b) *Trapdoor Oracle* $\mathscr{O}_T(w, pk)$. Given a keyword w and a public key pk, the challenger computes the trapdoor T_w by running the algorithm trapdoor $(\mathrm{sk}_R, \mathrm{pk}, w)$ and returns the trapdoor to $\mathscr{A}$

(iii) *Challenge*. When phase 1 ends, the adversary $\mathscr{A}$ outputs two challenge keywords w_0^* and w_1^*, which have not been queried to the oracles $\mathscr{O}_C(\cdot, \mathrm{pk}_R)$ and $\mathscr{O}_T(\cdot, \mathrm{pk}_S)$ before. Now, the challenger chooses a random bit $b \in \{0, 1\}$, computes the $T_{w_b^*} \longleftarrow \mathrm{trapdoor}(\mathrm{sk}_R, \mathrm{pk}_S, w_b^*)$, and returns it to the adversary $\mathscr{A}$

(iv) *Phase 2*. In this phase, the adversary $\mathscr{A}$ can continue to access the oracles, with the restriction that neither w_0^* nor w_1^* could be queried to the oracles $\mathscr{O}_C(\cdot, \mathrm{pk}_R)$ and $\mathscr{O}_T(\cdot, \mathrm{pk}_S)$

(v) *Guessing*. Finally, the adversary $\mathscr{A}$ outputs a bit b' as the guess of b. If $b' = b$, we say that $\mathscr{A}$ wins the game

We define $\mathscr{A}$'s advantage in breaking the TI security of PAEKS as

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{TI}}(\lambda) = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right|. \tag{3}$$

Definition 5 (CI security game). Similarly, the CI security game can be described as follows:

(i) *Initialization*. Given a security parameter λ, the challenger generates the global system parameter Param. Then, the challenger generates the receiver's public/-secret keys $(\mathrm{pk}_R, \mathrm{sk}_R)$ and the sender's public/secret keys $(\mathrm{pk}_S, \mathrm{sk}_S)$. It executes the adversary $\mathscr{A}$ on input $(\mathrm{Param}, \mathrm{pk}_S, \mathrm{pk}_R)$

(ii) *Phase 1*. The adversary $\mathscr{A}$ is allowed to adaptively query the following two oracles polynomial times:

(a) *Cipher Text Oracle* $\mathscr{O}_C(w, pk)$. Given a keyword w and a public key pk, the challenger computes the cipher text C by running the algorithm PAEKS $(\mathrm{sk}_S, \mathrm{pk}_R, w)$ and returns the cipher text to $\mathscr{A}$

(b) *Trapdoor Oracle* $\mathscr{O}_T(w, pk)$. Given a keyword w and a public key pk, the challenger computes the trapdoor T_w by running the algorithm trapdoor $(\mathrm{sk}_R, \mathrm{pk}_S, w)$ and returns the trapdoor to $\mathscr{A}$

(iii) *Challenge*. When phase 1 ends, the adversary $\mathscr{A}$ outputs two challenge keywords w_0^* and w_1^*, which have not been queried to the oracles $\mathscr{O}_C(\cdot, \mathrm{pk}_R)$ and $\mathscr{O}_T(\cdot, \mathrm{pk}_S)$ before. Now, the challenger chooses a random bit $b \in \{0, 1\}$, computes the $C_{w_b^*} \longleftarrow \mathrm{PAEKS}(\mathrm{sk}_S, \mathrm{pk}_R, w_b^*)$, and returns it to the adversary $\mathscr{A}$

(iv) *Phase 2*. In this phase, the adversary $\mathscr{A}$ can continue to access the oracles, with the restriction that neither w_0^* nor w_1^* could be queried to the oracles $\mathscr{O}_C(\cdot, \mathrm{pk}_R)$ and $\mathscr{O}_T(\cdot, \mathrm{pk}_S)$

(v) *Guessing*. Finally, the adversary $\mathscr{A}$ outputs a bit b' as the guess of b. If $b' = b$, we say that $\mathscr{A}$ wins the game

We define $\mathscr{A}$'s advantage in breaking the CI security of PAEKS as

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{CI}}(\lambda) = \left|\Pr\left[b' = b\right] - \frac{1}{2}\right|. \tag{4}$$

If for any PPT adversary $\mathscr{A}$, both $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{TI}}(\lambda)$ and $\mathrm{Adv}_{\mathscr{A}}^{\mathrm{CI}}(\lambda)$ are negligible in the security parameter λ; we say that the PAEKS is semantically secure against inside keyword guessing attacks.

3. Our PAEKS Scheme

In this section, we introduce a PAEKS scheme for an electronic medical record system. The system framework is given in Figure 2.

3.1. The Construction. Our PAEKS scheme is described as follows:

(i) *Setup* (λ). Select a cyclic group G with prime order p and a random generator g of G. Select three pseudorandom functions: $E : \mathscr{K}' \times \mathscr{W} \longrightarrow \{0, 1\}^n$, $f : \mathscr{K}'' \times$

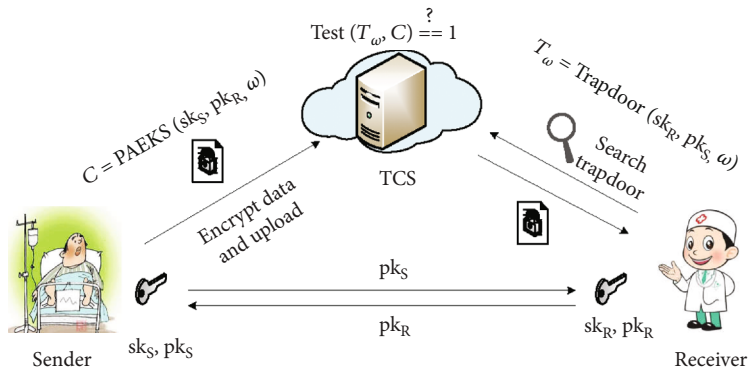

Figure 2: Our PAEKS system framework.

$\{0,1\}^n \longrightarrow \mathscr{K}$, and $F : \mathscr{K}' \times \{0,1\}^{m-n} \longrightarrow \{0,1\}^m$, where $\mathscr{K}'$, $\mathscr{K}''$, and $\mathscr{K}$ are the key spaces of the three PRFs, respectively, and $\mathscr{W}$ is the keyword space. Let H be a hash function, defined as $H : G \longrightarrow \mathscr{K}' \times \mathscr{K}''$. Finally, return $\text{Param} = (G, g, p, E, f, F)$

(ii) $KeyGen_S$ ($Param$). Randomly select $x \longleftarrow Z_p$, and set $\text{pk}_S := g^x$ and $\text{sk}_S := x$. Return sk_S and pk_S

(iii) $KeyGen_R$ ($Param$). Randomly select $y \longleftarrow Z_p$, and set $\text{pk}_R := g^y$ and $\text{sk}_S := y$. Return sk_R and pk_R

(iv) $PAEKS$ (sk_S, pk_R, w). To encrypt a keyword $w \in \mathscr{W}$, do the following:

(a) Compute the keys $k' \| k'' = H(\text{pk}_R^{\text{sk}_S}) = H(g^{xy})$

(b) Compute $X = E_{k'}(w)$ and $k = f_{k''}(X)$

(c) Select a random string $S \in \{0,1\}^{n-m}$ and set $U = S \| F_k(S)$

(d) Set $C = X \oplus U$

(e) Finally, return C

(v) $Trapdoor$ (sk_R, pk_S, w). Compute $k' \| k'' = H(\text{pk}_R^{\text{sk}_S}) = H(g^{xy}), X = E_{k'}(w)$ and $k = f_{k''}(X)$. Return the trapdoor $T_w = X \| k''$

(vi) $Test$ (T_w, C). Compute $U = C \oplus X$ and parse it as $S \| T$. If $T = F_k(S)$ holds, return 1; otherwise, return 0

3.1.1. Correctness. Let the receiver's key pair be $(\text{pk}_R, \text{sk}_R) = (g^x, x)$ and the sender's key pair be $(\text{pk}_S, \text{sk}_S) = (g^y, y)$. Then, the key $k' \| k'' = H(g^{xy})$ can be generated by each other. Let C be a cipher text of keyword w generated by the sender and T_w be the corresponding search trapdoor generated by the receiver. According to the keyword encryption algorithm, there must exist two strings X and U and a random string $S \in \{0,1\}^{n-m}$ such that $C = X \oplus U$, $X = E_{k'}(w)$, and $U = S \| F_k(S)$, where $k = f_{k''}(X)$. For a right trapdoor of keyword w, it should be in the form $T_w = X \| k$, where $X = E_{k'}(w)$ and $k = f_{k''}(X)$. So, $X \oplus C = U$. Let S be the first $n - m$ bits of U and T be the last m bits. Clearly, $T = F_k(S)$ will hold. Thus, for the same keyword, the cipher text will match with the corresponding trapdoor.

Fixing a cipher text C' of a distinct keyword $w' \neq w$, we have $C' = X' \oplus U'$, for some $X' = E_{k'}(w')$. Since E is a pseudorandom function, then $X = E_{k'}(w)$ is a random string over $\{0,1\}^n$ with probability at least $1 - \epsilon_E$. In this case, $X \oplus C'$ will be a random string. Since F is also a pseudorandom function, for a random string $S' \| T'$, the equation $F_k(S') = T'$ holds with probability at most $\epsilon_E + (1/2^n)$. Thus, the cipher text C' matches with the search trapdoor with a negligible probability. So our PAEKS scheme satisfies the correctness.

3.2. Security Proof. In this section, we prove that our PAEKS scheme satisfies both trapdoor indistinguishability and cipher text indistinguishability. Its trapdoor indistinguishability follows from the theorem below.

Theorem 6. *If the oracle Diffie-Hellman assumption holds and E is a pseudorandom function, then our PAEKS scheme achieves trapdoor indistinguishability. Specifically, for any PPT adversary $\mathscr{A}$, we have*

$$Adv_{\mathscr{A}}^{TI}(\lambda) \leq \epsilon_{odh} + \epsilon_E, \tag{5}$$

where ϵ_{odh} and ϵ_E are the advantages to break the ODH assumption and the pseudorandomness of the PRF E.

Proof. Let $\mathscr{A}$ be any PPT adversary that aims to break the security of trapdoor indistinguishability of our PAEKS scheme. We prove Theorem 6 by a sequence of games. Let Suc_i denote the event that $\mathscr{A}$ succeeds (i.e., $b' = b$) in the i-th game.

Game 0. This is the original trapdoor in a distinguishability game as defined in Definition 4. In this game, the challenger generates two public/secret key pairs $(\text{pk}_S, \text{sk}_S)$ and $(\text{pk}_R, \text{sk}_R)$ for the sender and the receiver, respectively, and gives the public keys to $\mathscr{A}$. In addition, the adversary can adaptively issue queries to the trapdoor oracle $\mathscr{O}_T(\cdot, \cdot)$ and cipher text oracle $\mathscr{O}_C(\cdot, \cdot)$ with any keyword $w \in \mathscr{W}$ and public key pk. But, for the two challenge keywords w_0^* and w_1^*,

the adversary cannot submit them to the oracles $\mathcal{O}_T(\cdot, \mathrm{pk_S})$ and $\mathcal{O}_C(\cdot, \mathrm{pk_R})$. Let $T_{w_b^*}$ denote the challenge trapdoor of w_b^*, where $b \longleftarrow \{0,1\}$. Let b' denote the guess of b by $\mathcal{A}$. So, $\mathcal{A}$'s advantage in this game is

$$\mathrm{Adv}_{\mathcal{A}}^{\mathrm{TI}}(\lambda) = \left|\Pr\left[\mathrm{Suc}_0\right] - \frac{1}{2}\right|, \tag{6}$$

Game 1. This game is the same as the previous game with the exception of $k' \| k''$ being sampled from $\mathcal{K}' \times \mathcal{K}''$ uniformly at random. Recall that in the previous game, the challenger computes $k' \| k''$ by $H(\mathrm{pk_S}^{\mathrm{sk_R}})$ (namely, $H(\mathrm{pk_R}^{\mathrm{sk_S}})$) according to the keyword encryption algorithm (namely, trapdoor generation algorithm). We now prove that

$$\left|\Pr\left[\mathrm{Suc}_1\right] - \Pr\left[\mathrm{Suc}_0\right]\right| \le \epsilon_{\mathrm{odh}}, \tag{7}$$

Given an instance of the oracle Diffie-Hellman problem (g, g^x, g^y, K), where $K = H(g^{xy})$ or K is a random string from $\mathcal{K}' \times \mathcal{K}''$, we construct an algorithm $\mathcal{B}$ to solve it using $\mathcal{A}$ as a subroutine. $\mathcal{B}$ sets $\mathrm{pk_S} := g^x$ and $\mathrm{pk_R} := g^y$ and gives them to $\mathcal{A}$. The corresponding secret keys are implicitly set to be x and y, respectively. In addition, $\mathcal{B}$ chooses the other system parameters, including E, f, F, by itself. Parse K as $k' \| k''$. When $\mathcal{A}$ issues queries to the oracles $\mathcal{O}_T(\cdot, \mathrm{pk})$ and $\mathcal{O}_C(\cdot, \mathrm{pk})$ with $\mathrm{pk} \notin \{\mathrm{pk_S}, \mathrm{pk_R}\}$, $\mathcal{B}$ involves the oracle $\mathcal{O}_{g^y}(\mathrm{pk})$ or $\mathcal{O}_{g^x}(\mathrm{pk})$ to obtain the shared key $k' \| k''$. When $\mathcal{A}$ issues queries to the oracles $\mathcal{O}_T(\cdot, \mathrm{pk_S})$ and $\mathcal{O}_C(\cdot, \mathrm{pk_R})$, $\mathcal{B}$ uses $k' \| k''$ to generate cipher texts and trapdoors. For example, for a keyword w, $\mathcal{B}$ computes the cipher text C as follows:

(1) Compute $X = E_{k'}(w)$ and $k = f_{k''}(X)$

(2) Select a random string $S \in \{0,1\}^{n-m}$ and set $U = S \| F_k(S)$

(3) Set $C = X \oplus U$

Given two challenge keywords w_0^* and w_1^*, $\mathcal{B}$ computes the challenge trapdoor $T_{w_b^*}$ as follows:

(1) Choose a random bit $b \in \{0,1\}$

(2) Compute $X^* = E_{k'}(w_b^*)$ and $k^* = f_{k''}(X^*)$

(3) Set $T_{w_b^*} = X^* \| k^*$

Finally, $\mathcal{A}$ outputs a bit b' as a guess of b. If $b' = b$, $\mathcal{B}$ outputs 1; otherwise, $\mathcal{B}$ outputs 0.

Clearly, if $K = H(g^{xy})$, the above game is identical to Game 0. Otherwise, it is identical to Game 1. So,

$$\begin{aligned} &\Pr\left[\mathcal{B}(g^x, g^y, H(g^{xy})) = 1 : x, y \longleftarrow \mathbb{Z}_p\right] = \Pr\left[\mathrm{Suc}_0\right], \\ &\Pr\left[\mathcal{B}(g^x, g^y, K) = 1 : x, y \leftarrow \mathbb{Z}_p, K \longleftarrow \mathcal{K}' \times \mathcal{K}''\right] = \Pr\left[\mathrm{Suc}_1\right]. \end{aligned} \tag{8}$$

This proves the result of Equation (7).

Game 2. This game is identical to the previous game with the exception of X^* being sampled randomly from $\{0,1\}^n$. Assuming that E is a pseudorandom function, we have

$$\left|\Pr\left[\mathrm{Suc}_2\right] - \Pr\left[\mathrm{Suc}_1\right]\right| \le \epsilon_E. \tag{9}$$

We now prove Equation (9). Given a challenge pseudorandom function E, we construct an algorithm $\mathcal{B}$ to break its pseudorandomness using $\mathcal{A}$ as a subroutine. $\mathcal{B}$ chooses the system parameter, the sender and receiver's public/secret key pairs, as in the previous game, with the exception of E being provided by its own challenger. Specifically, the random string $k'' \in \mathcal{K}''$ is chosen by $\mathcal{B}$ itself, but k' is implicitly defined by the secret key of the challenge pseudorandom function E. Next, we show how $\mathcal{B}$ answers $\mathcal{A}$'s queries of cipher texts and trapdoors with $(w, \mathrm{pk_R})$ or $(w, \mathrm{pk_S})$, respectively. For a keyword w, $\mathcal{B}$ computes its cipher text as follows:

(1) Query the challenger of E with w to obtain the result $X = E_{k'}(w)$

(2) Compute $k = f_{k''}(X)$

(3) Select a random string $S \in \mathcal{S}$ and compute $U = S \| F_k(S)$

(4) Set $C = X \oplus U$

$\mathcal{B}$ computes its trapdoor as follows:

(1) Submit w to its own challenger to obtain the result $X = E_{k'}(w)$

(2) Compute $k = f_{k''}(X)$

(3) Set $T_w = X \| k$

When $\mathcal{A}$ submits two challenge keywords w_0^* and w_1^*, $\mathcal{B}$ picks a random bit b and sends w_b^* to the oracle of PRF E for challenging. The PRF challenger will return the challenge PRF value X^* to $\mathcal{B}$, which may be $E_{k'}(w_b^*)$ or a random value. $\mathcal{B}$ then computes $k^* = f_{k''}(X^*)$ and returns $T_{w_b^*} = X^* \| k^*$ to the adversary. Finally, $\mathcal{A}$ outputs a guess bit b'. If $b' = b$, $\mathcal{B}$ outputs 1; otherwise, it outputs 0.

From the above analysis, it is clear that if $X^* = E_{k'}(w_b^*)$, $\mathcal{B}$ actually simulates an environment of Game 1 for the adversary $\mathcal{A}$. If X^* is random, the simulated environment is identical to Game 2. Thus, if $\mathcal{A}$'s success probability between Game 1 and Game 2 has difference $\Pr[\mathrm{Suc}_2] - \Pr[\mathrm{Suc}_1]$, then $\mathcal{B}$ can distinguish $X^* = E_{k'}(w_b^*)$ from a random one with the same advantage. This computes the proof of Equation (9).

Note that in Game 2, the challenge trapdoor is independent of the two challenge keywords. So, the adversary has no success advantage in this game, i.e.,

$$\Pr\left[\mathrm{Suc}_2\right] = \frac{1}{2}. \tag{10}$$

Table 1: Efficiency comparison.

Schemes	Encryption	Trapdoor	Test	IKGA	Multiuser
Boneh et al. [4]	$3E + H_1 + H_2 + P$	$E + H_1$	P	No	Yes
Huang and Li [16]	$3E + H_1$	$E + H_1 + P$	$2P$	Yes	No
Noroozi and Eslami [18]	$3E + H_1$	$E + H_1 + P$	$2P$	Yes	Yes
Qin et al. [19]	$3E + H_1 + H_2 + P$	$2E + H_1$	P	Yes	Unknown
Ours	$E + H_2 + 3F$	$E + H_2 + 2F$	F	Yes	Yes

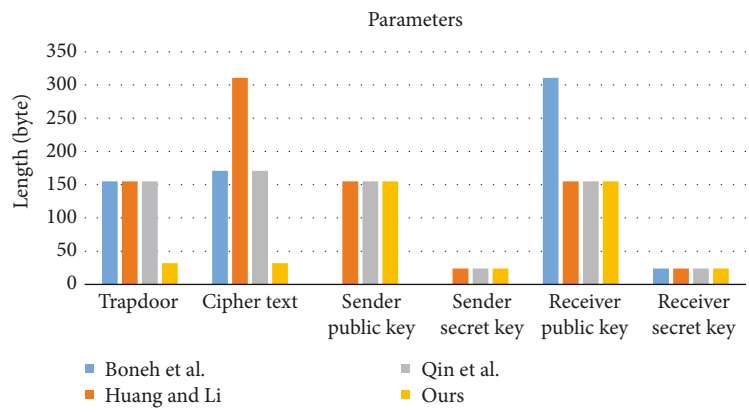

Figure 3: Comparison of parameter sizes.

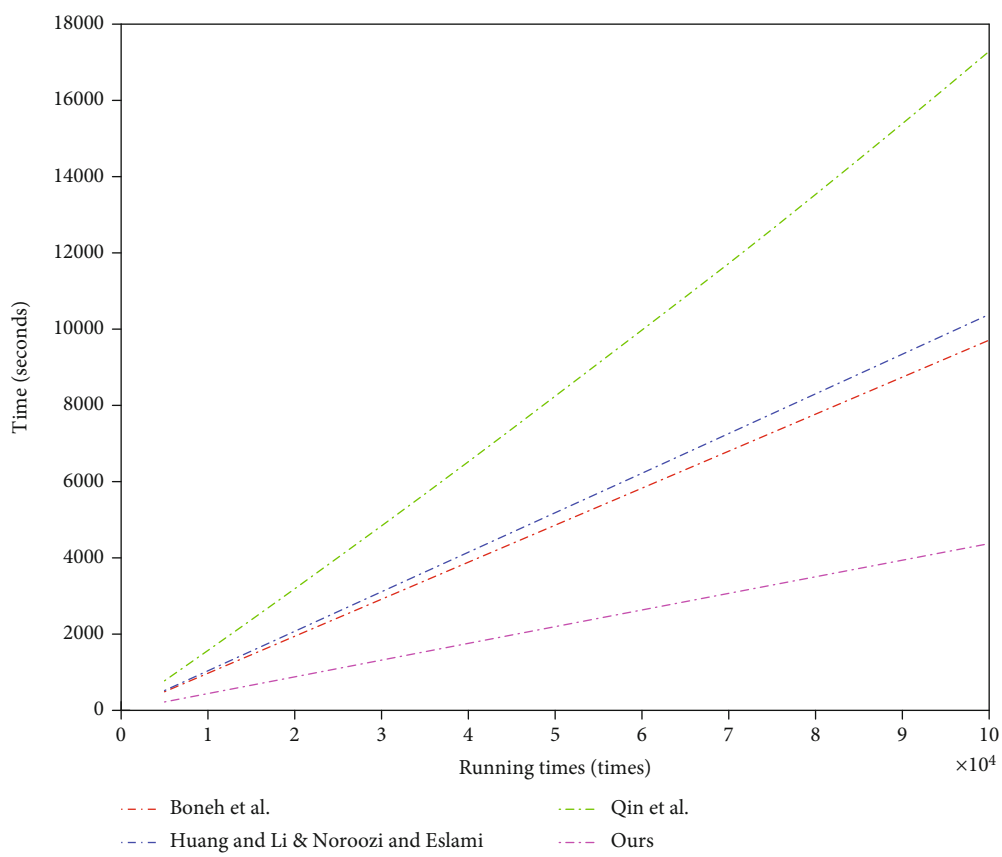

Figure 4: Running time of the encryption algorithm.

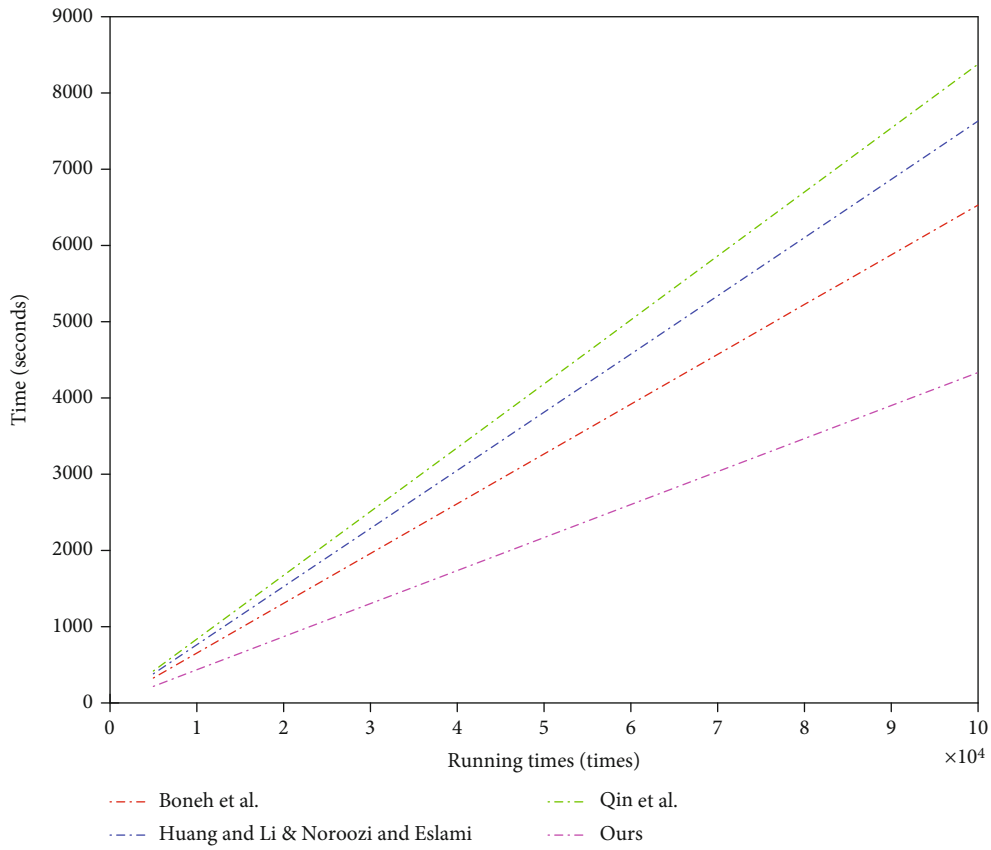

Figure 5: Running time of the trapdoor algorithm.

Taking Equations (6) to (10) together, it follows that

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{TI}}(\lambda) \leq \epsilon_{\mathrm{odh}} + \epsilon_E. \tag{11}$$

This completes the proof of Theorem 6.

The cipher text indistinguishability of our PAEKS scheme follows from the theorem below.

Theorem 7. *If the oracle Diffie-Hellman assumption holds and f, F are pseudorandom functions, then our PAEKS scheme achieves cipher text indistinguishability. Specifically, for any PPT adversary $\mathscr{A}$, we have*

$$Adv_{\mathscr{A}}^{CI}(\lambda) \leq \epsilon_{odh} + \epsilon_f + \epsilon_F, \tag{12}$$

where ϵ_{odh}, ϵ_f, and ϵ_F are the advantages to break the ODH assumption and the pseudorandomness of the PRFs f and F, respectively.

Proof. Similar to the proof of Theorem 6, we prove the above theorem also via a sequence of games. In each game, $\mathscr{A}$ is a PPT adversary, aiming to break the cipher text indistinguishability of our PAEKS scheme. b is the challenge random bit, selected by the challenger, and b' is $\mathscr{A}$'s guess bit. We denote the event that $b' = b$ in each game as Suc_i.

Game 0. This is the original cipher text indistinguishability game as defined in Definition 5. So,

$$\mathrm{Adv}_{\mathscr{A}}^{\mathrm{CI}}(\lambda) = \left|\Pr\left[\mathrm{Suc}_0\right] - \frac{1}{2}\right|. \tag{13}$$

Game 1. This game is the same as Game 0, except that the value $k' \| k''$ is chosen randomly from $\mathscr{K}' \times \mathscr{K}''$. Under the ODH assumption, these two games are computationally indistinguishable, i.e.,

$$\left|\Pr\left[\mathrm{Suc}_1\right] - \Pr\left[\mathrm{Suc}_0\right]\right| \leq \epsilon_{\mathrm{odh}}. \tag{14}$$

The proof of the above equation is similar to that of Equation (7); we omit it here.

Game 2. This game is identical to Game 1, except the following modification to the challenge cipher text. Suppose that w_b^* is the challenge keyword and $X^* = E_{k'}(w_b^*)$ is the corresponding internal value of the cipher text. In this game, k^* is selected randomly from $\mathscr{K}$, instead of being computed via $k^* = f_{k''}(X^*)$. Note that, for normal keyword cipher text, k is still computed from $f_{k''}(X)$. Under the assumption that f is a pseudorandom function, these two games are computationally indistinguishable. Specially, we have

$$\left|\Pr\left[\mathrm{Suc}_2\right] - \Pr\left[\mathrm{Suc}_1\right]\right| \leq \epsilon_f. \tag{15}$$

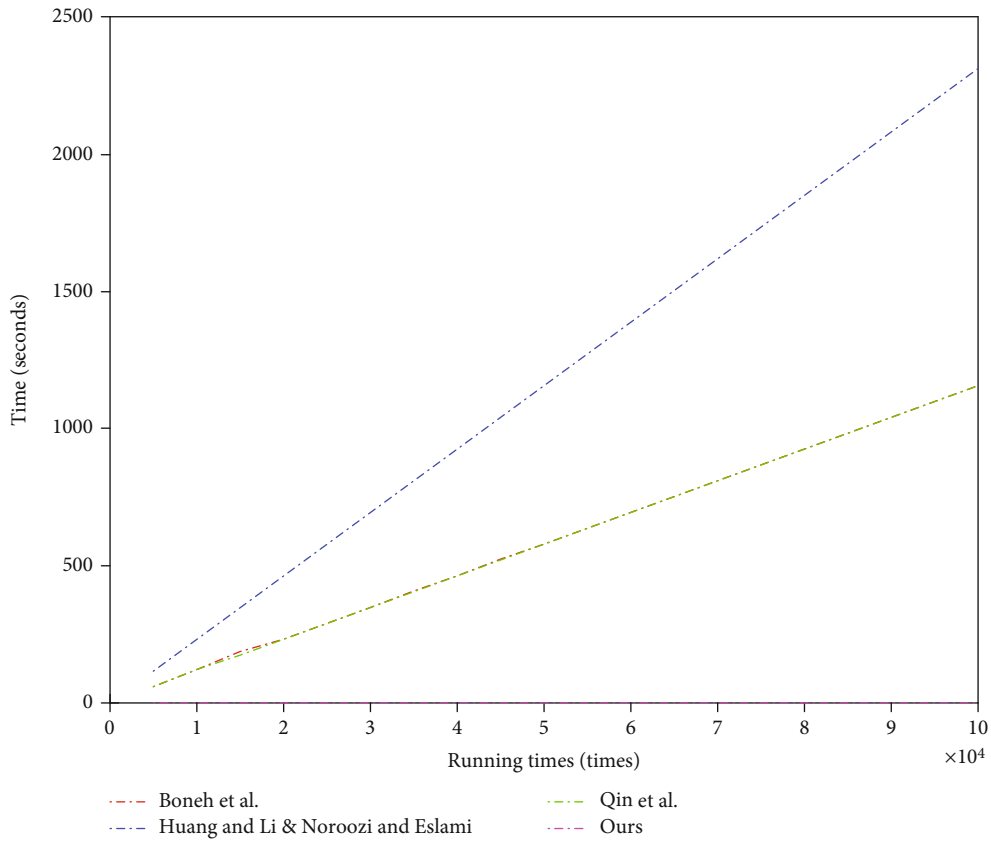

FIGURE 6: Running time of the test algorithm.

The proof of the above equation is similar to that of Equation (9); we omit it here.

Game 3. In this game, we replace the challenge value $U^* = S^* \| F_{k^*}(S^*)$ with a random string $U^* \longleftarrow \{0,1\}^n$. Recall that, in this game, k^* is sampled uniformly from $\mathscr{K}$. By the pseudorandomness of PRF F, $S^* \| F_{k^*}(S^*)$ is computationally indistinguishable from a random n-bit string. Similarly, we can prove that

$$|\Pr[\mathrm{Suc}_3] - \Pr[\mathrm{Suc}_2]| \leq \epsilon_F. \tag{16}$$

In Game 3, U^* is random and is independent of the challenge keywords. So, the adversary has no advantage in this game, i.e.,

$$\Pr[\mathrm{Suc}_3] = \frac{1}{2}. \tag{17}$$

Taking Equations (13) to (17) together, we complete the proof of Theorem 7.

From Theorems 6 and 7, we conclude that our PAEKS scheme is semantically secure against inside keyword guessing attack assuming that the ODH problem is hard and E, f, F are PRFs.

4. Experiments and Efficiency Comparison

In this section, we analyze the efficiency of our PAEKS scheme and compare it with some other related schemes, including Boneh et al.'s PEKS scheme [4] and PAEKS schemes of [16, 18, 19]. Except our scheme, all the others are designed in bilinear groups. That is, besides group G, there are another group G_T and a bilinear map e defined from $G \times G$ to G_T.

Table 1 demonstrates the theoretical result of efficiency comparison in terms of keyword encryption, trapdoor generation, testing, and two security properties. In the table, we use symbols "E" and "P" to denote the evaluation of a modular exponentiation and a bilinear pairing, respectively. "H_1" denotes a special hash function that maps an arbitrary string to a group element, while "H_2" denotes a traditional hash function, e.g., MD5. We denote the pseudorandom function as "F."

Figure 3 shows the length of each parameter in different PEKS/PAEKS schemes. With the exception of Boneh et al.'s scheme, the other three schemes involve the sender's public key and secret key in the keyword encryption algorithm and trapdoor generation algorithm, respectively. It can be seen from the figure that our scheme has shorter trapdoor and cipher text than other schemes. For the other parameters, our scheme still has comparable length with other schemes.

Among these operations, the computation of the pairing is usually the most time-consuming. According to the construction of H_1 in [21], its computation is usually inefficient

with the comparison of the traditional hash function. In a random oracle model, it is easy to construct a PRF from an efficient hash function. From these observations, we can see that our keyword testing algorithm should be much faster than that of the other three schemes. For encryption and the trapdoor generation, the advantage of our scheme is not obvious among them. In terms of security, Boneh et al.'s scheme cannot resist against IKGA. The scheme of [16] can prevent IKGA, but it is not secure in a multiuser setting. The scheme of [19] did not show its security in a multiuser setting.

To evaluate the efficiency of these schemes in practice, we use a laptop with 1.7 GHz Intel i3 CPU, 2 GB memory, and a Windows 7 operating system to implement them. We use the jPBC library and choose a type A pairing, which makes use of the curve $y^2 = x^3 + x$ over the field $\mathbb{F}_q$ for prime $q \equiv 3 \mod 4$. We run each algorithm with different times and record their time in seconds. The results are shown in Figures 4, 5, and 6, respectively. As the computations of Noroozi and Eslami and Huang and Li, they possess the same experimental results. Experiment results show that our encryption algorithm and trapdoor generation algorithm are slightly faster than those of the other schemes. But our keyword testing algorithm is significantly faster than that of the other schemes.

5. Conclusion

In this paper, we proposed a new public-key authenticated encryption scheme with keyword search. Our scheme uses the idea of the Diffie-Hellman key exchange protocol to generate a shared secret key between the sender and the receiver. The shared key can be viewed as the secret key of a symmetric-key searchable encryption scheme to encrypt keywords by the sender or to generate search trapdoors by the receiver. Under the ODH assumption, our PAEKS scheme can achieve both trapdoor indistinguishability and cipher text indistinguishability, and hence, it can resist inside keyword guessing attacks. The scheme is also efficient. Specifically, its keyword searching algorithm is very fast in the sense that it requires only one computation of PRF, while the previous schemes require at least one expensive pairing operation.

Data Availability

The data used to support the findings of this study are embedded in the programming. They are available from the corresponding author upon request (email: qinbaodong@xupt.edu.cn).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (grant numbers 61872292 and 61772418), the Key Research and Development Program of Shaanxi (grant number 2020ZDLGY08-04), and the Basic Research Program of Qinghai Province (grant number 2020-ZJ-701).

References

[1] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: a survey," *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 978–996, 2018.

[2] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Security and privacy in the medical internet of things: a review," *Security and Communication Networks*, vol. 2018, Article ID 5978636, 9 pages, 2018.

[3] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*, pp. 44–55, Berkeley, CA, USA, May 2000.

[4] D. Boneh, G. di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004*, C. Cachin and J. Camenisch, Eds., vol. 3027 of Lecture Notes in Computer Science, , pp. 506–522, Springer, 2004.

[5] H. S. Rhee, W. Susilo, and H. J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electronics Express*, vol. 6, no. 5, pp. 237–243, 2009.

[6] L. Fang, W. Susilo, C. Ge, and J. Wang, "Public key encryption with keyword search secure against keyword guessing attacks without random oracle," *Information Sciences*, vol. 238, pp. 221–241, 2013.

[7] P. Xu, H. Jin, Q. Wu, and W. Wang, "Public-key encryption with fuzzy keyword search: a provably secure scheme under keyword guessing attack," *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2266–2277, 2013.

[8] C.-H. Wang and T.-Y. Tu, "Keyword search encryption scheme resistant against keyword-guessing attack by the untrusted server," *Journal of Shanghai Jiaotong University (Science)*, vol. 19, no. 4, pp. 440–442, 2014.

[9] Z. Y. Shao and B. Yang, "On security against the server in designated tester public key encryption with keyword search," *Information Processing Letters*, vol. 115, no. 12, pp. 957–961, 2015.

[10] Y. Wu, X. Lu, J. Su, and P. Chen, "An efficient searchable encryption against keyword guessing attacks for sharable electronic medical records in cloud-based system," *Journal of Medical Systems*, vol. 40, no. 12, article 258, 2016.

[11] M. Ma, D. He, N. Kumar, K. K. R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 759–767, 2018.

[12] W. C. Yau, R. C. W. Phan, S. H. Heng, and B. M. Goi, "Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester," *International Journal of Computer Mathematics*, vol. 90, no. 12, pp. 2581–2587, 2013.

[13] C. Li, C. Lee, C. Weng, T. Wu, and C. Chen, "Cryptanalysis of an efficient searchable encryption against keyword guessing attacks for shareable electronic medical records in cloud-based system," in *Information Science and Applications 2017-ICISA 2017, Macau, China, 20-23 March 2017*, K. Kim and N. Joukov, Eds., vol. 424 of Lecture Notes in Electrical Engineering, , pp. 282–289, Springer, 2017.

[14] Y. Lu, G. Wang, and J. Li, "Keyword guessing attacks on a public key encryption with keyword search scheme without random oracle and its improvement," *Information Sciences*, vol. 479, pp. 270–276, 2019.

[15] T. Y. Wu, C. M. Chen, K. H. Wang, and J. M. T. Wu, "Security analysis and enhancement of a certificateless searchable public key encryption scheme for IIoT environments," *IEEE Access*, vol. 7, pp. 49232–49239, 2019.

[16] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403-404, pp. 1–14, 2017.

[17] T.-Y. Wu, C.-M. Chen, K.-H. Wang, J. M.-T. Wu, and J.-S. Pan, "Security analysis of a public key authenticated encryption with keyword search scheme," in *Recent Advances in Intelligent Information Hiding and Multimedia Signal Processing: Proceeding of the Fourteenth International Conference on Intelligent Information Hiding and Multimedia Signal Processing, November, 26-28, 2018, Sendai, Japan, Volume 1*, pp. 178–183, Springer, 2019.

[18] M. Noroozi and Z. Eslami, "Public key authenticated encryption with keyword search: revisited," *IET Information Security*, vol. 13, no. 4, pp. 336–342, 2019.

[19] B. Qin, Y. Chen, Q. Huang, X. Liu, and D. Zheng, "Public-key authenticated encryption with keyword search revisited: security model and constructions," *Information Sciences*, vol. 516, pp. 515–528, 2020.

[20] M. Abdalla, M. Bellare, and P. Rogaway, "The oracle Diffie-Hellman assumptions and an analysis of DHIES," *Topics in Cryptology — CT-RSA 2001: The Cryptographers' Track at RSA Conference 2001 San Francisco, CA, USA, April 8–12, 2001 Proceedings*, D. Naccache, Ed., , pp. 143–158, Springer, 2001.

[21] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 586–615, 2003.