On Security of a Secure Channel Free Public Key Encryption with Conjunctive Field Keyword Search Scheme

Yang Lu, Gang Wang, Jiguo Li
College of Computer and Information, Hohai University, No.8, Focheng Xi Road, Jiangning District, Nanjing, China
e-mails: luyangnsd@163.com, wg15951977612@163.com, lijiguo@hhu.edu.cn

Corresponding author: luyangnsd@163.com

Public key encryption with keyword search is a practical cryptographic paradigm that enables one to search for the encrypted keyword without compromising the security of the original data. Recently, Hwang et al. proposed a secure channel free public key encryption with conjunctive field keyword search (SCF-PECKS) scheme and claimed that their scheme can withstand the keyword guessing attack and does not need the secure channel. In this paper, by presenting three concrete attacks, we demonstrate that Hwang et al.’s SCF-PECKS scheme fails to achieve the security against keyword guessing attacks by either outsider attackers or malicious insider servers. The presented attacks show that Hwang et al.’s scheme is vulnerable to the keyword guessing attack regardless of whether the trapdoors are sent via public channel or secure channel. Therefore, devising a secure SCF-PECKS scheme remains an unsolved problem until now.

KEYWORDS: public key encryption, conjunctive field keyword search, secure channel free, keyword guessing attack.

1. Introduction

With the development of cloud computing technology, the amount of sensitive data to be stored on the cloud servers is rapidly increasing. By using the cloud storage, the users can get the convenient service and greatly reduce the cost of local data management. To protect the confidentiality of the data, more and more companies and individuals choose to encrypt their data before outsourcing them to the cloud storage servers. By using the encryption techniques, the sensitive data are transformed into the random strings
and are not readable to anyone except the holder of the corresponding decryption key. Thus, the data confidentiality is guaranteed. However, the application of traditional encryption techniques makes it difficult for the cloud storage server to selectively retrieve the encrypted data. A common solution is to allow the user to download all his/her encrypted data from the cloud storage server and then search for them whenever he wants to retrieve the data. In this way, the user has to download and decrypt all encrypted data, regardless of what data he/she is searching for. Obviously, this solution is extremely inefficient due to the high cost of network transmission and the heavy overhead at the user devices.

In order to solve the ciphertext retrieval problem, the notion of keyword search over encrypted data (i.e., searchable encryption) was introduced [26, 2]. In [26], Song et al. proposed the first searchable encryption scheme in symmetric cryptography. Searchable encryption enables the user to search encrypted data without revealing the underlying plaintexts and the searched keywords. Therefore, it provides a promising solution to address the ciphertext retrieval in clouds [9-11, 29]. Public key encryption with keyword search (PEKS) was proposed by Boneh et al. [2] in 2004. In Boneh et al.’s proposal, a PEKS scheme contains three entities: sender, receiver and server (also called tester). The sender encrypts the data with a standard public key encryption (PKE) scheme and then appends a PEKS ciphertext of a keyword to the encrypted data. To retrieve the encrypted data with keyword ciphertext on the server, the receiver sends a trapdoor of the searched keyword to the server. After receiving the trapdoor, the server can test whether the keyword associated with the encrypted data is the same as the keyword encoded in the trapdoor without revealing any information about the searched keyword and the original data. Finally, the server returns the encrypted data containing the searched keyword to the receiver.

Following Boneh et al.’s work [2], searchable PKE has attracted great attention from the research community and lots of PEKS schemes [8, 12, 14, 18, 19, 28] have been presented. To convey the trapdoors securely, PEKS needs a secure channel between the server and each receiver. However, it is well known that building a secure channel is very expensive. To solve the problem, Baek et al. [1] designed a new framework that removes the secure channel between the server and the receiver, which is called secure channel free public key encryption with keyword search (SCF-PEKS) or designed server public key encryption with keyword search (dPEKS). The basic idea of SCF-PEKS is to make the server generate and keep its own public key and private key. When the sender wants to produce a keyword ciphertext, he/she must take the server’s public key as input in the keyword encryption algorithm. Thus, only the designed server is able to execute the testing algorithm by using its private key. After Baek et al.’s first construction of SCF-PEKS, a number of SCF-PEKS schemes have been proposed [6, 13, 15, 22, 24, 25, 31]. Searchable encryption was also introduced into attribute-based encryption so as to associate the keyword ciphertexts and the trapdoors with sets of attributes [20, 21, 27]. To provide multi-keyword search, Park et al. [23] first proposed a public key encryption with conjunctive field keyword search (PEKS) scheme. Subsequently, several PEKS schemes were proposed, e.g. [3, 4, 7, 17, 30]. Recently, Hwang et al. [16] showed that two previous PEKS schemes [7, 30] are insecure under the keyword guessing attack from the outsider attacker. To remove the requirement of secure channel in PEKS, Hwang et al. [16] extended the framework of PEKS and proposed a secure channel free PEKS (SCF-PEKS) scheme. They asserted that their scheme resists the keyword guessing attack and does not require conveying the trapdoors via secure channel.

In [5], Byun et al. first observed the keyword guessing attacks on some PEKS schemes. By performing this attack, an attacker may reveal the keyword encoded in a keyword ciphertext or a trapdoor. The keyword guessing attack exploits the low-entropy property of the commonly-used keywords. In practice, the users usually select some keywords from a small keyword space (such as an English dictionary) to generate the keyword ciphertexts and the trapdoors. Therefore, the attacker is able to traverse the keyword space to guess a correct keyword in an acceptably short time. As introduced in [5], the latest edition of Merriam-Webster’s Collegiate Dictionary includes approximately 225 000 entries. Thus, to guess a correct keyword, the probability is about \(1/2^{225}\). For another example, the current Oxford English Dictionary contains about 600 000 entries and thus the probability to guess the correct keyword is about \(1/2^{10}\). Actually, this pro-
bability will be higher if the attack applies to some particular application, e.g. email application, which usually has a small keyword space containing some commonly-used keywords such as {urgent, high, normal, low, ...}. Undoubtedly, the keyword guessing attack has become the most devastating attack on the keyword search encryption schemes, since it may lead to the disclosure of the information pertaining to the encrypted data.

In this paper, we demonstrate that Hwang et al.’s SCF-PECKS scheme [16] is vulnerable to the keyword guessing attack. In [16], Hwang et al. asserted that their scheme guards against the keyword guessing attack and has the property of no secure channel. However, by giving three concrete keyword guessing attacks, we show that either the outsider attacker or the insider server is able to launch the keyword guessing attack on their scheme to disclose the keywords encoded in the encrypted data or the trapdoor. The presented attacks indicate that Hwang et al.’s SCF-PECKS scheme is insecure under the keyword guessing attack regardless of whether the trapdoors are sent via public channel or secure channel.

The rest of our paper is organized as follows. In Section 2, we briefly review some background preliminaries. In Section 3, we present three keyword guessing attacks on Hwang et al.’s SCF-PECKS scheme. Finally, we draw our conclusions in Section 4.

### 2. Preliminaries

#### 2.1. Bilinear Pairing

Assume that $q$ is a big prime number, $G_1$ and $G_2$ are two cyclic groups of the same order $q$. The bilinear pairing is a map $e: G_1 \times G_1 \rightarrow G_2$ which satisfies the following three attributes:

1. **Bilinearity**: $e(xU, yV) = e(U, V)^{xy}$ for all $U, V \in G_1$ and $x, y \in \mathbb{Z}_q$.
2. **Non-degeneracy**: There exists $U, V \in G_1$ such that $e(U, V) \neq 1$.
3. **Computability**: There exists an efficient algorithm to compute $e(U, V)$ for all $U, V \in G_1$.

#### 2.2 The Definition of a SCF-PECKS Scheme

A SCF-PECKS scheme consists of the following algorithms [16]:

1. **Setup($\lambda$)**: This algorithm takes a security parameter $\lambda$ as input and generates a set of global parameters $\mathcal{GP}$.
2. **KenGen\textsubscript{Server}($\mathcal{GP}$)**: This algorithm takes the global parameters $\mathcal{GP}$ as input and generates a public/private key pair $(pk_s, sk_s)$ for a server $S$.
3. **KeyGen\textsubscript{Receiver}($\mathcal{GP}$)**: This algorithm takes the global parameters $\mathcal{GP}$ as input and generates a public/private key pair $(pk_r, sk_r)$ for a receiver $R$.
4. **dPECKS($\mathcal{GP}$, $pk_s$, $pk_r$, $D$)**: This algorithm takes the global parameters $\mathcal{GP}$, a server’s public key $pk_s$, a receiver’s public key $pk_r$, and a keyword set $D$ as input and outputs a dPECKS ciphertext $C$.
5. **dTrapdoor($\mathcal{GP}$, $pk_s$, $sk_r$, $Q$)**: This algorithm takes the global parameters $\mathcal{GP}$, a server’s public key $pk_s$, a receiver’s private key $sk_r$ and a query $Q$ as input and outputs a trapdoor $T_r$.
6. **dTest($\mathcal{GP}$, $sk_r$, $C$, $T_r$)**: This algorithm takes the global parameters $\mathcal{GP}$, a server’s private key $sk_r$, a dPECKS ciphertext $C$ and a trapdoor $T_r$ as input, returns “yes” if $w_i = w'_i$, $i = 1, ..., t$, or “no” otherwise.

#### 2.3. Adversary Types

As introduced in [5], two kinds of adversaries are considered in our keyword guessing attacks. They are outsider attacker and malicious insider server.

1. **Outsider attacker**: This adversary is able to eavesdrop on the public channel and obtain the ciphertexts and the trapdoors that are conveyed over the public channel. However, this adversary cannot perform the dTest algorithm, because the dTest algorithm requires the server’s private key as input.
2. **Malicious insider server**: This adversary is able to receive the encrypted data and keyword ciphertexts from the senders. It can also receive the trapdoors from the receivers. Most importantly, it is able to execute the dTest algorithm to test whether a keyword ciphertext and a trapdoor correspond to a same keyword set by using its private key.

### 3. Keyword Guessing Attacks on Hwang et al.’s SCF-PECKS Scheme

In this section, we first briefly review Hwang et al.’s SCF-PECKS scheme [16] and then present three keyword guessing attacks on it. The presented attacks...
show that Hwang et al.’s scheme is insecure against keyword guessing attacks by either outsider attackers or insider servers.

3.1. A Review of Hwang et al.’s Scheme

Hwang et al.’s SCF-PECKS scheme works as follows:

1 **Setup(λ):** Let $G_1$ be an additive group of prime order $p$ with a generator $g$ and $G_2$ be a multiplicative group of same order $p$. Let $e: G_1 \times G_1 \rightarrow G_2$ be a bilinear pairing and $H: \{0, 1\}^* \rightarrow Z_p^*$ be a secure one-way hash function. Let $K_{S_0}$ denote the keyword space. This algorithm outputs the global parameters $GP = \{G_1, G_2, e, g, H, K_{S_0}\}$.

2 **KeyGen_{server}(GP):** This algorithm selects a random value $a \in Z_p^*$ and sets $sk_a = a$. Then, it computes $pk_a = (pk_a^{G_1}, pk_a^{G_2}) = (g^a, g^a)$. Finally, it outputs the server’s public/private key pair $(pk_a, sk_a)$.

3 **KeyGen_{receiver}(GP):** This algorithm selects a random value $\beta \in Z_p^*$ and sets $sk_B = \beta$. Then, it computes $pk_B = (pk_B^{G_1}, pk_B^{G_2}) = (g^\beta, g^\beta)$. Finally, it outputs the receiver’s public/private key pair $(pk_B, sk_B)$.

4 **dPECKS(GP, pk_{gs}, pk_{gs}, D):** In this algorithm, the sender encrypts $m$ keywords with the server’s public key $pk_{gs} = (pk_{gs}^{G_1}, pk_{gs}^{G_2})$ and the receiver’s public key $pk_{gs} = (pk_{gs}^{G_1}, pk_{gs}^{G_2})$. This algorithm first chooses a random value $r \in Z_p^*$ and computes $C_1 = H(w_1) \cdot pk_{gs}^{G_1}$, $C_2 = H(w_2) \cdot pk_{gs}^{G_1}$, $C_m = H(w_m) \cdot pk_{gs}^{G_1}$, $C_{m+1} = rg$ and $C_{m+2} = pk_{gs}^{G_2} \cdot pk_{gs}^{G_1}$. It then outputs the keyword ciphertext $C_k = (C_1, C_2, C_m, C_{m+1}, C_{m+2})$.

5 **dTrapdoor(GP, pk_{gs}, sk_{gs}, Q):** In this algorithm, an authorized receiver produces a trapdoor for the keywords $w_i, w_{i+1}, ..., w_n$ in the query $Q$. This algorithm first selects a random value $k \in Z_p^*$ and computes $V = \sum_{i=1}^n H(w_i)$. Then, it computes $T_1 = kg$ and $T_2 = (sk_k + V)^{-1} \cdot (pk_{gs})^k$. Finally, it outputs the trapdoor $T_w = (T_1, T_2, I_1, I_2, ..., I_n)$.

6 **dTest(GP, sk_{gs}, C, T_w):** Once receiving a trapdoor $T_w$ from the receiver, the server executes this algorithm to verify whether the trapdoor $T_w$ matches with the keyword ciphertext $C_k$ by using its private key $sk_{gs}$. This algorithm first computes $(u, \bar{u})$ from the keyword ciphertext $C_k$ as follows:

\[
\begin{align*}
  u &= (C_1 \times C_2 \times ... \times C_n) / e(C_{m+1}, g)^{sk_{gs}} = \prod_{i=1}^n H(w_i), \\
  \bar{u} &= (C_{m+2}) / e(C_{m+1}, g)^{sk_{gs}} = pk_{gs}^{G_2}.
\end{align*}
\]

Then, it computes $Z = \frac{T_2}{e(T_1, g)} = (\beta + V)^{-1}$ and checks if $e(u \cdot \bar{u}^*, g^*) = e(g, g)$. If so, the output “yes” and “no” otherwise.

3.2. Keyword Guessing Attacks by Outsider Attackers

In Hwang et al.’s SCF-PECKS scheme, the keyword ciphertext generated by the server’s scheme will be secure against the receiver’s public keys while the trapdoor is generated with the server’s public key and the receiver’s private key. It seems that the scheme is secure against keyword guessing attacks by outsider attackers, since the execution of the dTest algorithm requires the input of the server’s private key. However, an outsider attacker can launch a keyword guessing attack on the scheme through the following steps:

**Step 1:** The attacker chooses a target receiver $R$. Then, it uses the target receiver $R$’s public key $pk_{R}$ and the server $S$’s public key $pk_{gs}$ to produce the keyword ciphertexts $C_{w_1}, C_{w_2}, ..., C_{w_n}$ for all candidate keyword sets $W_1, W_2, ..., W_n$ of its choice, where $W_i = \{w_i, w_{i+1}, ..., w_n\}$ for each $i \in \{1, 2, ..., n\}$.

**Step 2:** The attacker simulates a sender by sending $Enc_{w_1}(\{C_{w_1}, C_{w_2}, ..., C_{w_n}\})$ to the target receiver $R$, where $Enc_{w_1}, Enc_{w_2}, ..., Enc_{w_n}$ denote the crafted encrypted documents corresponding to the keyword sets $W_1, W_2, ..., W_n$, respectively. After that, the encrypted data $Enc_{w_1}(\{C_{w_1}, C_{w_2}, ..., C_{w_n}\})$ will be transmitted to and stored on the server $S$.

**Step 3:** When receiving a trapdoor $T_w$ from the receiver $R$, the server $S$ returns all matched encrypted documents to the receiver $R$ based on the result of the dTest algorithm. The returned documents may contain some encrypted documents faked by the attacker previously.

**Step 4:** Since a SCF-PECKS scheme does not assume a secure channel, the outsider attacker is able to wiretap the communication between the server $S$ and the receiver $R$. Therefore, it knows which trapdoor has been sent by the receiver $R$. Upon observing the returned encrypted documents, including one of its crafted encrypted documents (e.g., $Enc_{w_1}$), the attacker can determine that the encoded keyword set is $W_i$, which implies a correct guess.

In the above attack, although the outsider attacker cannot execute the dTest algorithm, it is able to make
use of the server as a testing oracle to get the testing results. This attack is feasible as it does not require a compromise of the server by just eavesdropping the communication between the server and the receiver.

3.3. Keyword Guessing Attacks by Insider Servers

The above attack can be avoided if the keyword trapdoors are sent to the designated server via secure channel. However, Hwang et al.’s SCF-PECKS scheme is still insecure against keyword guessing attacks by insider servers.

3.3.1. Attack I

A malicious insider server $S$ can launch a keyword guessing attack on Hwang et al.’s scheme through the following steps:

**Step 1:** It obtains a valid trapdoor $T_w$ from any receiver $R$.

**Step 2:** It guesses a candidate keyword set $W'$ and then produces a keyword ciphertext $C$ for the keyword set $W'$ with the receiver $R$’s public key $pk_R$ and its public key $pk_S$.

**Step 3:** Using its private key $sk_s$, it executes the $dTest$ algorithm to verify whether the keyword ciphertext $C$ matches with the trapdoor $T_w$. If it does, then $W'$ is a correct keyword set. Otherwise, it goes to Step 2 and continues its guessing.

3.3.2. Attack II

A malicious insider server $S$ also can execute a keyword guessing attack on Hwang et al.’s scheme through the following steps:

**Step 1:** It obtains a valid trapdoor $T_w = (T_1, T_2, I_1, I_2, ..., I_d)$ from any receiver $R$.

**Step 2:** It guesses a candidate keyword set $W' = \{w'_1, w'_2, ..., w'_t\}$ and then computes the hash value $H_s(w'_i)$ of the each keyword $w'_i$. After that, it calculates $V = \prod_{i=1}^{t} H_s(w'_i)$.

**Step 3:** Using its private key $sk_s$ and the value $V$, it verifies whether the trapdoor $T_w = (T_1, T_2, I_1, I_2, ..., I_d)$ satisfies the equation $g^{r_1} = pk_{s2} \cdot g^{r_2}$. If the equation holds, then $W'$ is a correct keyword set. Otherwise, it goes to Step 2 and continues its guessing.

Obviously, an insider server is able to reveal the keyword set encoded in a trapdoor by the above attack. What is worse, is that after guessing the correct keyword set, the insider server can further run the $dTest$ algorithm to determine which encrypted document contains the keyword set.

4. Conclusions

In this paper, we have shown that Hwang et al.’s SCF-PECKS scheme [16] is insecure against keyword guessing attacks. Our first attack on Hwang et al.’s scheme indicates that an outsider attacker is able to make use of the server as a testing oracle to verify the correctness of its guesses. This attack is feasible due to the communication between the receiver and server is over a public channel. Therefore, a secure channel is still required for a SCF-PECKS scheme to guard against the keyword guessing attacks by outsider attackers. However, our other two attacks show that this is still not enough to protect a SCF-PECKS scheme from the keyword guessing attacks by insider servers. It seems that it is impossible to devise a scheme against keyword guessing attacks under the current framework of SCF-PECKS. Therefore, it is necessary to design a new framework for SCF-PECKS.

Acknowledgments

We would like to present our thanks to the anonymous reviewers for their helpful comments. This work is supported by the National Natural Science Foundation of China under Grant Nos. 61772009 and U1736112, the Natural Science Foundation of Jiangsu Province under Grant No. BK20161511, the Fundamental Research Funds for the Central Universities under Grant Nos. 2016B10114 and 2017B17014.

References


20. Li, J., Lin, X., Zhang, Y., Han, J. KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. IEEE Transactions on


