A New Trapdoor-indistinguishable Public Key Encryption with Keyword Search

Yuanjie Zhao
Department of Mathematics
Xidian University, China
xdzhaoyuanjie@163.com

Xiaofeng Chen*
School of Telecommunications Engineering
Xidian University, China
xfchen@xidian.edu.cn

Hua Ma
Department of Mathematics
Xidian University, China
ma_hua@126.com

Qiang Tang
DIES, Faculty of EEMCS
University of Twente, the Netherlands
q.tang@utwente.nl

Hui Zhu
School of Telecommunications Engineering
Xidian University, China
zhuhui@xidian.edu.cn

Abstract

The public key encryption with keyword search (PEKS) provides a way for users to search data which are encrypted under the users' public key on a storage system. However, the original schemes are based on the unrealistic assumption of a secure channel between the receiver and the server. Baek et al. [1] first proposed a secure channel-free public key encryption with keyword search (SCF-PEKS) to remove the assumption. However, Rhee et al. [2] point out that the SCF-PEKS scheme suffers from the keyword-guessing attack and proposed a scheme which satisfies the property of trapdoor indistinguishability without using an additional secure channel. In this paper, we construct a new efficient trapdoor-indistinguishable public key encryption with keyword search. Moreover, we prove that the proposed scheme can achieve the desired security properties.

Keywords: Searchable encryption, Keyword search, Trapdoor, PEKS

1 Introduction

With the development of Cloud computing, more and more private and confidential information is being centralized into cloud [3][4]. Therefore, people are increasingly concerned about the security of the cloud. Traditionally, in order to protect the sensitive information stored on the cloud, some kind of access control mechanism may be applied in the database management system, such as MS Access. Access control is an effective way to protect your information under the assumption of a trusted server on which the database runs. However, access control is not a panacea in the real-world applications. In some cases, we cannot fully trust the server. An alternative solution is the encryption the data before saving them on the cloud. This can achieve the confidentiality of the data even against the inside attackers such as curious database system administrators. What the user need to do is to keep the encryption keys carefully without revealing them to the internal system manager. However, an inherent problem is how to retrieve the encrypted data efficiently.

*Corresponding author: Xidian University, Xi'an 710071, P.R.China, Tel: +86-29-88204749

Keyword search over encrypted data which enables a user to search encrypted data without leaking any information on the query and data can resolve this problem. Let us consider the following application: Suppose Alice is a decision-makers of a company, and many employees of the company send Alice emails which contains a lot of decision-making information. Assume that an employee encrypts his emails using a standard public key system and appends to the resulting ciphertext of a Public Key Encryption with Keyword Search (PEKS) of each keyword in a could service. In order to send a message M with keywords $W_1, \dots, W_k$, an employee sends

$$E_{A_{pub}}(M) \parallel PEKS(A_{pub}, S_{pub}, W_1) \parallel \cdots \parallel PEKS(A_{pub}, S_{pub}, W_k)$$

to the sever, where A_{pub} is Alice's public key, and S_{pub} is Server's public key. Alice gives the server a trapdoor T_W via the public channel and then the server can use T_W to select the desired emails [5].

In the original scheme, a certain keyword corresponds to a constant trapdoor. Therefore, an attacker who intercepts and captures the communications can statistics the frequency of occurrences of these trapdoors, and then he can choose the highest frequency trapdoor to attack [6]. Once the attack is successful, an attacker may know Alice's privacy interests. To solve this problem, the proposed PEKS scheme should satisfy the following properties: 1) Removing secure channel: this means that the trapdoors can be sent via a public channel and an outside attacker cannot determine which PEKS ciphertext encrypts which keyword even the attacker gets all the trapdoors for the keywords; 2) Trapdoor indistinguishability: the notion is firstly introduced in [2], which guarantees that the trapdoor does not reveal any information on any keyword without the server's private key.

In this paper, we construct a new trapdoor-indistinguishable public key encryption with keyword search, which does not require a secure channel between the receiver and the server. The trapdoor of our scheme is updated every time, that means an attacker can not distinguish two trapdoors even the two trapdoors come from the same keyword. It can effectively prevent information leakage. Compared with the existing schemes, the proposed scheme has advantages of better performance from the view of the receiver.

1.1 Related Work

With the increasing private data having been stored on the sever, the questions about how to protected them [7][8][9][10][11] and how to search them [12][13][14][15] have become a focus for researchers. There are three different types of keyword search systems over encrypted data, and the solutions for each are different. The first type is a public storage system, a user wants to store his sensitive information to a remote database and wishes never leak any information to the remote database administrator. Later, the user can retrieve information from the remote database with a particular keyword. Solutions to this problem has been presented in the early 1990's by Ostrovsky [11] and Ostrovsky and Goldreich [**?**]. A secure keyword search scheme by using a symmetric cipher was first proposed in [16], then a conjunctive keyword search scheme that provide a solution for users to search the conjunction of multiple keywords with one encrypted query was proposed in [17]. The second type is a vendor system. Here the database data is public, but the information stored on the database is public and the user is unaware of the data stored on the database. At the same time, the user wishes to search for the information without revealing to the database administrator. Public Information Retrieval (PIR) protocols offer a solution for this problem [18] [19]. An oblivious keyword search scheme, which makes a user can retrieve data that contain a keyword, was introduced in [20]. The scheme was later improved in [21]. The third type is a store-and-forward system, the user can search information that is encrypted under the user's public key on a storage system. A public key encryption with keyword searching (PEKS) scheme was first proposed in [5]. However, the scheme assumes a secure channel between the receiver and the server. Baek et al. [1] first proposed a secure channel-free public key encryption with keyword search (SCF-PEKS) to solve this problem. An efficient searchable public key encryption scheme was proposed in [22] and a fuzzy keyword search scheme was proposed in [23].

However, there are still some security flaws in SCF-PEKS scheme, as pointed out in [6]. Given a trapdoor, an attacker can determine the trapdoor corresponds to which keyword. Therefore, if an attacker who intercepts and captures the trapdoors between the receiver and the server, he is able to know the user's private interests and querying patterns by analyzing the frequency of occurrences of trapdoors and keyword-guessing attack. Rhee et al. [2] first addressed this problem and introduced the notion of "trapdoor indistinguishability" to solve it.

1.2 Organization

The organization of this paper is as follows. Some preliminaries are given in Section 2. The proposed efficient trapdoor-indistinguishable public key encryption with keyword search and its security analysis are given in Section 3. Finally, conclusions will be made in Section 4.

2 Preliminaries

In this section we first introduce some related definitions and then overview the SCF-PEKS scheme proposed in [1].

2.1 Overview of the SCF-PEKS scheme

As described in [1], three parties called "sender", "receiver" and "server" are involved. The sender is a party that creates and sends encrypted keywords, which we call "PEKS ciphertexts". The server is a party that receives PEKS ciphertexts and performs search upon receiving trapdoors from the receiver. The receiver is a party that creates trapdoors and sends them to the server to find the data that it wants. We review the formal definition of SCF-PEKS given in [1], the SCF-PEKS scheme consists of the following algorithms:

- A common Parameter Generation Algorithm $KeyGen_{Param}(k)$: Taking a security parameter $k \in \mathbb{N}$ as input, this algorithm generates a common parameter cp.
- A Server Key Generation Algorithm $KeyGen_{Server}(cp)$: Taking a common parameter cp as input, this algorithm generates a private and public key pair (sk_S, pk_S) of the receiver.
- A Receiver Key Generation Algorithm $KeyGen_{Receiver}(cp)$: Taking a common parameter cp as input, this algorithm generates a private and public key pair (sk_R, pk_R) of the receiver.
- A Secure Channel Free PEKS Algorithm $SCF-PEKS(cp, pk_S, pk_R, w)$: Taking a common parameter cp, a server's public key pk_S, a receiver's public key pk_R and a keyword w as input, this algorithm returns a PEKS ciphertext S which is a searchable encryption of w. We write $S = SCF-PEKS(cp, pk_S, pk_R, w)$.
- A Trapdoor Generation Algorithm $Trapdoor(cp, sk_R, w)$: Taking a common parameter cp, a receiver's private key sk_R and a keyword w as input, this algorithm generates a trapdoor T_w for w.
- A Test Algorithm $Test(cp, T_w, sk_S, S)$: Taking a common parameter cp, a trapdoor T_w for the keyword w, the server's private key sk_S, and a PEKS ciphertext $S = SCF-PEKS(pk_S, pk_R, w')$, this algorithm returns a symbol "Correct" if $w = w'$ and "Incorrect" otherwise.

The SCF-PEKS scheme works as follows: 1) The receiver and the server run the *KenGen* algorithm to get their public/private key pairs, respectively. 2) When the sender wants to send an email m containing keywords $W_1 \ldots, W_k$ to the server, The sender first runs the E algorithm to encrypt the email, and then runs SCF-PEKS to encrypt all the keywords, respectively, and sends both the ciphertext of the email and keywords to the server. 3) When the receiver wants to retrieve emails containing keyword $W_j (j \in \mathbb{Z}^+)$, he runs the $Trapdoor$ algorithm to get $W_j^{\prime}s$ trapdoor T_{W_j}, and sends it to the server. 4) When receiving the trapdoor, the server runs the $Test$ algorithm to decide whether a given email contains keyword W_j, if the email exists, the server sends the email to the receiver. otherwise, the server return 0 to the receiver.

2.2 Security of our scheme

We define security for our scheme in the sense of IND-CPA security (*Beak et al.*2008) and Trapdoor-IND-CPA security [2]. As described in [5], IND-CPA guarantees that the server that has not obtained the trapdoors for given keywords cannot tell which PEKS ciphertext encrypts which keyword, and the outside attacker that has not obtained the server's private key cannot make any decisions about the PEKS ciphertexts even though the attacker gets all the trapdoors for the keywords that it holds. As described in [2], Trapdoor-IND-CPA asks that an attacker (excluding the server and the receiver) can not distinguish between the trapdoors of two challenge keywords. We define the IND-CPA security and trapdoor-IND-CPA for our scheme as follows, which had been described in [1] [2]:

Let A be an attacker whose running time is bounded by t which is polynomial in a security parameter k and B be a challenger. We consider the following three games:

Game 1: A is assumed to be a server.

Phase 1-1: The common parameter generation algorithm $KeyGen_{Param}(k)$, the two key generation algorithms $KeyGen_{Receiver}(k)$ and $KeyGen_{Server}(k)$ are run. A common parameter cp, private and public key pairs of the receiver and the server, which we denote by (sk_R, pk_R) and (sk_S, pk_S) respectively, are then generated. cp, pk_R, sk_S, and pk_S are given to A while sk_R is kept secret from A.

Phase 1-2: A queries a number of keywords, each of which is denoted by w, to the trapdoor generation oracle Trapdoor and obtains a corresponding trapdoor T_w.

Phase 1-3: A outputs a target keyword pair (w_0^*, w_1^*). (Notice that none of w_0^* nor w_1^* has been queried for obtaining a corresponding trapdoor in Phase 1-2). Upon receiving this, the PEKS oracle PEKS chooses $\beta \in \{0,1\}$ uniformly at random and creates a target $PEKS$ ciphertext $S^* = PEKS(cp, pk_S, pk_R, w_\beta^*)$ and returns it to A.

Phase 1-4: A issues a number of trapdoor extraction queries as in Phase 1-2. The restriction here is that w_0^* and w_1^* are not allowed to be queried as trapdoor extraction queries.

Phase 1-5: A outputs its guess $\beta \in \{0,1\}$.

We define A's success in Game 1 by $Succ_{Game1}(k) = Pr[\beta' = \beta] - \frac{1}{2}$.

Game 2: A is assumed to be an outside attacker (including the receiver).

'

Phase 2-1: The common parameter generation algorithm $KeyGen_{Param}(k)$, the two key generation algorithms $KeyGen_{Receiver}(k)$ and $KeyGen_{Server}(k)$ are run. A common parameter cp, private and public key pairs of the receiver and the server, which we denote by (sk_R, pk_R) and (sk_S, pk_S) respectively, are then generated. cp, pk_R, sk_R, and pk_S are given to A while sk_S is kept secret from A.

Phase 2-2: A queries a number of keywords, each of which is denoted by w to the trapdoor generation oracle Trapdoor and obtains a corresponding trapdoor T_w.

Phase 2-3: A outputs a target keyword pair (w_0^*, w_1^*). Upon receiving this, the PEKS oracle PEKS chooses $\beta \in \{0,1\}$ uniformly at random and creates a target PEKS ciphertext $S^* = PEKS(cp, pk_S, pk_R, w_\beta^*)$ and returns it to A.

Phase 2-4: A issues a number of trapdoor extraction queries as in Phase 2-2. Differently from Game 1, w_0^* and w_1^* are allowed to be queried as trapdoor extraction queries.

Phase 2-5: A outputs its guess $\beta' \in \{0,1\}$.

We define A's success in Game 2 by $Succ_{Game2}(k) = Pr[\beta' = \beta] - \frac{1}{2}$. The scheme is said to be IND-CPA secure if $Succ_{IND-CPA}(k) \triangleq Succ_{Gamei}(k)$, where i is either 1 or 2, is negligible in k.

Game 3: A is assumed to be an outside attacker.

Phase 3-1: The common parameter generation algorithm $KeyGen_{Param}(k)$, the two key generation algorithms $KeyGen_{Receiver}(k)$ and $KeyGen_{Server}(k)$ are run. A common parameter cp, private and public key pairs of the receiver and the server, which we denote by (sk_R, pk_R) and (sk_S, pk_S) respectively, are then generated. cp, pk_R, sk_R, and pk_S are given to A while sk_S is kept secret from A.

Phase 3-2: A makes the trapdoor queries of the form w, and B can adaptively ask T_w for any keyword w.

Phase 3-3: A outputs a target keyword pair (w_0^*, w_1^*), on which it wishes to be challenged. The restrictions are that both w_0^* and w_1^* has been queried for obtaining the corresponding trapdoors, T_{w_1} and T_{w_2}, and that the attacker did not previously ask for the trapdoors, T_{w_1} and T_{w_2}, in phase3-2. B picks a random $\beta \in \{0,1\}$ and returns T_{w_β} to A.

Phase 3-4: A makes trapdoor queries of the form, w, and B can adaptively ask Tw for any keyword w, as long as $w \neq w_0, w_1$.

Phase 3-5: A outputs its guess $\beta' \in \{0,1\}$ and win the Game 3, if $\beta = \beta'$.

The advantage of A in breaking trapdoor indistinguishability in our scheme is defined as $Succ_{trapdoor-ind-cpa}(k) = |Pr[\beta = \beta'] - 1/2|$. The scheme is said to be Trapdoor-IND-CPA secure if $Succ_{trapdoor-ind-cpa}(k)$ is negligible.

3 A New Trapdoor Indistinguishable Public Key Encryption with Keyword Search

3.1 Theoretical Background

Let $\mathbb{G}_1$ be a cyclic additive group generated by P, whose order is a prime q, and $\mathbb{G}_2$ be a cyclic multiplicative group of the same order q. Let a and b be elements of $\mathbb{Z}_q$. A bilinear pairing is a map e: $\mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ with the following properties:

1. Bilinear: $e(aR, bQ) = e(R,Q)^{ab}$ for all $R, Q \in \mathbb{G}_1$ and all $a, b \in \mathbb{Z}_q$.
2. Computable: There is a polynomial time algorithm to compute $e(R,Q) \in \mathbb{G}_2$, for any $R, Q \in \mathbb{G}_1$.
3. Non-degenerate: There exists R and $Q \in \mathbb{G}_1$, such that $e(R,Q) \neq 1$.

In the following, we introduce some problems in $\mathbb{G}_1$.

- Discrete Logarithm Problem (DLP): Given two elements P and Q, to find an integer $n \in \mathbb{Z}_q$, such that $Q = nP$ whenever such an integer exists.

- Computation Diffie-Hellman Problem (CDHP): Given P, aP, bP for a, $b \in \mathbb{Z}_q$, to compute abP.

Since the DDHP in the group $\mathbb{G}_1$ is easy, it can not be used to design cryptosystems in $\mathbb{G}_1$. Boneh and Franklin introduced a new problem in $(\mathbb{G}_1, \mathbb{G}_2, e)$ named Bilinear Diffie-Hellman Problem:

- Bilinear Diffie-Hellman Problem (BDHP): Given P, aP, bP, cP for a, b, $c \in \mathbb{Z}_q$, to compute $e(P,P)^{abc} \in \mathbb{G}_2$.

Trivially, the BDHP in $(\mathbb{G}_1, \mathbb{G}_2, e)$ is no harder than the CDHP in $\mathbb{G}_1$ or $\mathbb{G}_2$. However, the converse is still an open problem. On the other hand, currently it seems that there is no polynomial time algorithm to solve the BDHP in $(\mathbb{G}_1, \mathbb{G}_2, e)$ with non-negligible probability. The security of our proposed identity-based chameleon hash scheme without key exposure is also based on the hardness of the BDHP in $(\mathbb{G}_1, \mathbb{G}_2, e)$.

3.2 Construction

The mean idea of our scheme is to find a way to prevent the attackers who can intercept and capture the trapdoors between the receiver and the server from attacking the trapdoors. The trapdoor of our scheme is updated every time, that means an outside attacker can not distinguish two trapdoors even the two trapdoors come from the same keyword. It can effectively prevent information leakage. At the same time, the scheme should guarantee the server can search data efficiently. Our scheme consists of the following algorithms:

- $KeyGen_{Param}(k)$: Generate a group $\mathbb{G}_1$ of prime order $q \geq 2^k$, a random generator P of $\mathbb{G}_1$, and construct a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$. Specify hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \mathbb{G}_2 \rightarrow \{0,1\}^k$. Return $cp = (q, \mathbb{G}_1, \mathbb{G}_2, e, P, H_1, H_2, d_W)$ as a common parameter, where d_W denotes a description of a keyword space.

- $KeyGenServer(cp)$: Choose $x \in \mathbb{Z}_q^*$ uniformly at random and compute $X = xP$. Choose $Q \in \mathbb{G}_1^*$ uniformly at random. Return $pk_S = (cp, Q, X)$ and $sk_S = (cp, x)$ as the server's public and private key respectively.

- $KeyGen_{Receiver}(cp)$: Choose $y \in \mathbb{Z}_q^*$ uniformly at random and compute $Y = yP$. Return $pk_R = (pk_S, Y)$ and $sk_R = (cp, y)$ as the receiver's public and private key respectively.

- $PEKS(cp, pk_S, pk_R, w)$: Choose $r \in \mathbb{Z}_q^*$ and compute $R = (U, V, t)$ such that $(U, V, t) = (rP, rY, t)$, where $t = e(H_1(w), rP)e(rQ, X)$ Return R as a PEKS ciphertext.

- $Trapdoor(cp,\ sk_R, w)$: Choose $\tilde{a} \in \{0,1\}^*$ at random Compute $T_{w1} = [y^{-1}H_1(w) + H_1(\tilde{a})] \oplus [H_1[e(yQ, xP)]$ and $T_{w2} = yH_1(\tilde{a}) \in \mathbb{G}_1$. Return $TW = (T_{w1}, T_{w2})$ as a trapdoor for w.

- $Test(cp, TW, sk_S, R)$: The server first computes $T_w = T_{w1} \oplus H_1[e(xQ, yP)]$, and computes $S = e(T_{w2}, U)$, $t' = e(xQ, U)^{-1}$ and $T = tt' = e(H_1(w), rP)$, finally, tests if $H_2[e(T_w, V)] = H_2(T \cdot S)$. If the equation holds return "Correct" and "Incorrect" otherwise.

Correctness: The scheme is correct when assuming the PEKS cipertext $R = [rP, rY, t]$, where $t = e(H_1(w_0), rP)e(rQ, X)$ is valid for w_0 and trapdoor $TW = (T_{w1}, T_{w2})$, where $T_{w1} = [y^{-1}H_1(w_1) + H_1(\tilde{a})] \oplus [H_1[e(yQ, xP)]$ and $T_{w2} = yH_1(\tilde{a})$.

Note that $T_w = T_{w1} \oplus H_1[e(xQ, yP)] = y^{-1}H_1(w_1) + H_1(\tilde{a})$, $S = e(T_{w2}, U) = e(yH_1(\tilde{a}), rP)$, $T = tt' = e(H_1(w_0), rP)e(rQ, X)e(xQ, U)^{-1} = e(H_1(w_0), rP)$.

We use the following equations to show correctness if the w_1 is identical to w_0:

$$\begin{aligned} H_2[e(T_w,V)] &= H_2[e(y^{-1}H_1(w_1)+H_1(\tilde{a}),rY)] \\ &= H_2[e(y^{-1}H_1(w_1),yrP)e(H_1(\tilde{a}),yrP)] \\ &= H_2[e(H_1(w_1),rP)e(yH_1(\tilde{a}),rP)] \\ &= H_2(T\cdot S) \end{aligned}$$

Remark. As an outside attacker, he can not calculate the value of $H_1[e(xQ,yP)]$, $H_1[e(yQ,xP)]$, $t' = e(xQ,U)^{-1}$ due to the difficulty of BDH problem. Therefore, he cannot test whether the equation $H_2[e(T_w,V)] = H_2(T\cdot S)$ holds or not. That means only the server can process the test as described above. Furthermore an outside attacker can not decide which PEKS ciphertext encrypts which keyword even though he gets all the trapdoors for the keywords that it holds. As a result our scheme satisfied the properties of "removing secure channel".

3.3 Comparison

In this section, we present a comparison of security and performance between our scheme and the others.

Table 1 Comparison of security assumption and properties.

Scheme	CT Ind	Trap Ind	SC
PEKS (prposed in [5])	Satisfied	Not satisfied	Required
SCF-PEKS (prposed in [1])	Satisfied	Not satisfied	Required
dPEKS (prposed in [2])	Satisfied	satisfied	Not required
Our scheme	Satisfied	satisfied	Not required

We use CT Ind, Trap Ind, and SC as abbreviations for the meaning of PEKS Ciphertext Indistinguishability, Trapdoor Indistinguishability, and Secure Channel between a receiver and server, respectively.

Table 2 Comparison of performance and efficiency.

Scheme	Size(CT)	Size (Trap)	Comp(Receiver)
PEKS (prposed in [5])	$2G$	G	e
SCF-PEKS (prposed in [1])	$2G$	G	e
dPEKS (prposed in [2])	$2G$	$2G$	3e+m
Our scheme	$3G$	$2G$	$2e$

Let Size(CT), Size(Trap), and Comp(Receiver) be the abbreviate for the Size of PEKS Ciphertext, Size of Trapdoor, Computation Cost of the receiver, respectively. Besides, we denote by G the bits of the element in $\mathbb{G}$. We denote by m and e the modular multiplication and modular exponentials, respectively.

3.4 Security Analysis

In this section, we show that our scheme satisfies the property of trapdoor indistinguishability .

Theorem 1. The new trapdoor-indistinguishable public key encryption with keyword search scheme satisfies the property of trapdoor indistinguishability.

Proof: As an outside attacker, he can not distinguish if two trapdoors were generated by the same keyword due to the following reasons: 1). The trapdoor of our scheme is updated every time due to the difference of string $\tilde{a}$ we selected. 2). Even if two trapdoors were generated by the same keyword, they still can not be identified. The reason is due to the confidentiality of the value of $H_1[e(xQ,yP)]$ or $H_1[e(yQ,xP)$, therefore, the value of $H_1[e(xQ,yP)]$ or $H_1[e(yQ,xP)]$ becomes very important to this scheme. Assuming the value of $H_1[e(xQ,yP)]$ or $H_1[e(yQ,xP)]$ was leaked, an attacker will determine if two trapdoors come from the same keyword. The reasons are as follows:

If an attacker know the value of $H_1[e(xQ,yP)]$ or $H_1[e(yQ,xP)]$, he can compute the value of $T_w = y^{-1}H_1(w)+H_1(\tilde{a})$. Now, he use the following steps to decide whether the two trapdoors were generated by the same keyword or not.

- STEP-1 $A_1 = e(y^{-1}H_1(w)+H_1(\tilde{a}_1),yP) = e(y^{-1}H_1(w),yP)e(yH_1(\tilde{a}_1),P)$.
- STEP-2 $A_2 = e(yH_1(\tilde{a}_1),P)^{-1}$.
- STEP-3 $A = A_1A_2 = e(y^{-1}H_1(w),yP) = e(H_1(w),P)$.

Therefore, if two trapdoors come from the same keyword, the value of $A_{w_0} = e(H_1(w_0),P)$ will equal to the value of $A_{w_1} = e(H_1(w_1),P)$, if $w_0 = w_1$. In this way, an attacker will determine if two trapdoors come from the same keyword. But suppose an outside attacker know $Q = zP \in \mathbb{G}_1$ where z is a element in $\mathbb{Z}_q^*$, the server's public key $xP \in \mathbb{G}_1$, where $x \in \mathbb{Z}_q^*$ is the server's private key and the receiver's public key $yP \in \mathbb{G}_1$, where $y \in \mathbb{Z}_q^*$ is the receiver's private key, he cannot calculate $e(P,P)^{xyz}$, assuming that the BDH problem is hard. Furthermore, he cannot compute the value of $H_1[e(xQ,yP)]$ or $H_1[e(yQ,xP)]$.

4 Conclusion

Since the public-key encryption keyword search scheme was first proposed in [5], most of the existing schemes were only concerned about the security of PEKS ciphertext. However, the security of trapdoor is equally important for users. In this paper, we construct a new efficient trapdoor-indistinguishable public key encryption with keyword search, which incorporates the advantages of removing secure channel and trapdoor indistinguishability. By comparison of security and performance between our scheme and the others, we show that our scheme is more efficient and practical when applied to a cloud environment.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 60970144), major national science and technology projects (No. 2012ZX03002003) and the Fundamental Research Funds for the Central Universities (Nos. K50510010003, K50511010001, JY10000901034 and K50510010030).

References

[1] J.Baek, R.Safiavi-Naini, and W.Susilo, "Public key encryption with keyword search revisited," in *Proc. of the 8th International Conference on Computational Science and Its Applications (ICCSA'08), Perugia, Italy, LNCS*, vol. 5072. Springer-Verlag, June–July 2008, pp. 1249–1259.

[2] H. Rhee, J. Park, W. Susulo, and D. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *Journal of Systems and Software*, vol. 83, no. 5, pp. 766–771, May 2010.

[3] P. Mell and T. Grance, "Draft nist working definition of cloud computing," http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, June 2009.

[4] M. Armbrust, "Above the clouds: A berkeley view of cloud computing," http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html, February 2009.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *Proc. of 2004 International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT’04), Interlaken, Switzerland, LNCS*, vol. 3027. Springer-Verlag, May-June 2004, pp. 506–522.

[6] I. Jeong, J. Kwon, D. Hong, and D. Lee, “Constructing peks schemes secure against keyword guessing attacks is possible?” *Computer Communications*, vol. 32, no. 2, pp. 394–396, February 2009.

[7] D.Boneh and M.Franklin, “Identity-based encryption from the weil pairing,” in *Proc. of the 21st Annual International Cryptology Conference (CRYPTO’01), Santa Barbara, California, USA, LNCS*, vol. 2139. Springer-Verlag, August 2001, pp. 213–229.

[8] F.Emekci, D.Agrawal, A.E.Abbadi, and A.Gulbeden, “Privacy preserving query processing using third parties,” in *Proc. of the 22nd International Conference on Data Engineering (ICDE’06), Atlanta, Georgia, USA*. IEEE, April 2006, pp. 27–27.

[9] X. Boyen and B. Waters, “Anonymous hierarchical identity based encryption (without random oracles),” in *Proc. of the 26th Annual International Cryptology Conference (Crypto’06), Santa Barbara, California, USA, LNCS*, vol. 4117. Springer-Verlag, August 2006, pp. 290–307.

[10] J. Camenisch and A. Lysyanskaya, “A signature schemes and anonymous credentials from bilinear maps,” in *Proc. of the 24th Annual International Cryptology Conference (Crypto’04), Santa Barbara, California, USA, LNCS*, vol. 3152. Springer-Verlag, August 2004, pp. 56–72.

[11] R. Ostrovsky, “Software protection and simulation on oblivious rams,” Ph.D. dissertation, University of California at Berkeley Computer Science Division, November 1993.

[12] Q. Liu, G. Wang, and J. Wu, “Secure and privacy preserving keyword searching for cloud storage services,” *Journal of Networks and Computer Applications*, vol. 35, no. 3, pp. 927–933, May 2012.

[13] H. Haclgiimfi, B. Iyer, LiC, and MehrotraS, “Executing sql over encrypted data in database-service-provider model,” Database Research Group at University of California, Tech. Rep. TR-DB-02-02, 2002.

[14] M. Bellare, A. Boldyreva, and A. O‘Neill, “Deterministic and efficiently searchable encryption,” in *Proc. of the 27th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO’07), Santa Barbara, CA, USA, LNCS*, vol. 4622. Springer-Verlag, August 2007, pp. 535–552.

[15] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, “Executing sql over encrypted data in the database-service-provider model,” in *Proc. of the 2002 ACM SIGMOD international conference on Management of data (SIGMOD’02), Madison, Wisconsin, USA*. ACM, June 2002, pp. 216–227.

[16] D. Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in *Proc. of the 2000 IEEE symposiumon security and privacy (S&P’00), Berkeley, California, USA*. IEEE, May 2000, pp. 44–55.

[17] P. Golle, J. Staddon, and B. Waters, “Secure conjunctive keyword search over encrypted data,” in *Proc. of the 2nd International Conference on Applied Cryptography and Network Security (ACNS’04), Yellow Mountain, China*, vol. 3089. Springer-Verlag, June 2004, pp. 31–45.

[18] E. Kushilevitz and R. Ostrovsky, “Replication is not needed: Single database,” in *Proc. of the 38th Annual Symposium on Foundations of Computer Science (FOCS’97), Miami Beach, Florida, USA*. IEEE, October 1997, pp. 364–373.

[19] S. Goldwasser and S. Micali, “Probabilistic encryption,” *Journal of Computer and system Sciences*, vol. 28, no. 2, pp. 270–299, April 1984.

[20] W.Ogata and K.Kurosawa, “Oblivious keyword search,” *Journal of Complexity*, vol. 20, no. 2, pp. 356–371, April-June 2004.

[21] H. S. Rhee, J. W. Byun, D. H. Lee, and J. Lim, “Oblivious conjunctive keyword search,” in *Proc. of the 6th International Workshop on Information Security Applications (WISA’05), Jeju Island, Korea, LNCS*, vol. 3786. Springer-Verlag, August 2005, pp. 318–327.

[22] D. J. Park, K. Kim, and P. J. Lee, “Public key encryption with conjunctive field keyword search,” in *Proc. of the 5th International Workshop on Information Security Applications (WISA’04), Jeju Island, Korea, LNCS*, vol. 3325. Springer-Verlag, August 2004, pp. 73–86.

[23] J. Li, Q. Wang, C. Wang, K. R. N. Cao, and W. Lou, “Fuzzy keyword search over encrypted data in cloud computing,” in *Proc. of the 29th conference on Information communications (INFOCOM’10), San Diego,*

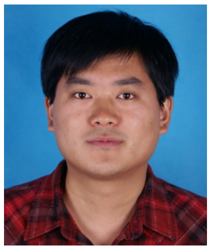
California, USA. IEEE, April 2010, pp. 1–5.

Yuanjie Zhao is a graduate student at the Faculty of Science, in Xidian University. His research interests include public key cryptography, cloud computing security and searchable encryption.

Xiaofeng Chen received his Ph.D. in cryptography from the Xidian University in 2003. He is currently a Professor at the School of Telecommunications Enginnering, Xidian University. His research interests include public key cryptography, financial cryptography, and cloud computing security.

Hua Ma recievered her Master's degree in Applied Mathematics from the Xidian University in 1990. She is currently a Professor at the Faculty of Science, Xidian University. Her research interests include public key cryptography, network security, and electronic commerce.

Qiang Tang is a postdoc researcher at the Distributed and Embedded Security Research Group in the Computer Science department at University of Twente, the Netherlands. He is leading the Kindred Spirits project and working on the security and privacy issues for online social networks and recommendation systems. Before this, he was a postdoc researcher in the Crypto group at Ecole Normale Superieure, Paris, France. He received his MSc degree from Peking University , China in 2002 and obtained his PhD degree from Royal Holloway, University of London , UK in 2007.



Hui Zhu received the Ph.D. degree in information security from Xidian University in 2009. Now, he works as an associate professor in Xidian University. His current research interests include network security and security authentication protocol.