Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (Student Contributions)

Yang Ma
London Metropolitan University
London, UK
(+44) 7746 250289
yam0209@my.londonmet.ac.uk

Hassan B Kazemian
London Metropolitan University
London, UK
+44 (0)20 7133 2933
kazemian@staff.londonmet.ac.uk

ABSTRACT

Public Key Encryption with Keyword Search (PEKS) enables users to search encrypted messages by a specific keyword without compromising the original data security. Traditional PEKS schemes allow users to search one keyword only instead of multiple keywords. Therefore, these schemes may not be applied in practice. Besides, some PEKS schemes are vulnerable to Keyword Guessing Attack (KGA). This paper formally defines a concept of Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (tSCF-MPEKS) and then presents a concrete construction of tSCF-MPEKS. The proposed scheme solves multiple keywords search problem and satisfies the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability. Its security is semantic security in the random oracle models under Bilinear Diffle-Hellman (BDH) and 1-Bilinear Diffie-Hellman Inversion (1-BDHI) assumptions so that it is able to resist KGA.

CCS CONCEPTS

• **Security and privacy → Cryptography**; **Public key (asymmetric) techniques**; **Public key encryption**;

KEYWORDS

Public Key Encryption with Keyword Search (PEKS), Trapdoor-indistinguishable, Keyword Guessing Attack (KGA), multiple keywords search

1 INTRODUCTION

Computer has played a pivotal role in social civilization and progress in the last several decades. Recently, computers have become more prevalent in connecting people as well as producing substantial benefits for society and enterprise. With the development of Internet, companies and people are willing to store their data into the third party (i.e. cloud servers) for saving local memory, reducing expenses and extra backups. But, keeping data into the third party may bring about some negative influences. The stored data may start to bear the brunt of any attack. For instance, crackers are delighted in launching port scanning to exploit the vulnerability of hosts and then intrude the victim's system without authentication and always ruin the operating systems in the end. Besides, some unfriendly hackers may capture the data packages on the network and then try to unpack these packages to obtain the information. Hacking brings huge lost both in money and energy. Therefore, many experts and technicians dedicate themselves to avoid these attacks to some extent. It is noticeable that Public Key Encryption with Keyword Search (PEKS) is one of the most advanced cryptographic systems to ensure data transmission security.

Boneh et al.[4] proposed the first PEKS scheme in 2004, which allows users to search encrypted messages by a specific keyword without compromising the security of the primitive data. This scheme is Indistinguishability under Chosen Plaintext Attack (IND-CPA) secure but has its limitations. For instance, it requires a secure channel between the server and the receiver, but building secure channel is much expensive and unrealistic in some cases. Hence, Baek et al.[1] came up with a new method to remove the secure channel from the original PEKS scheme, namely "Secure Channel Free Public Key Encryption with Keyword Search (SCF-PEKS)". In reality, the keyword for searching is limited and may suffer Keyword Guessing Attack (KGA). Byun et al.[5] were first found that PEKS was compromising from off-line KGA. Tang et al.[12] introduced a new PEKS scheme resisting off-line KGA, but the encryption algorithm is much complex. Later, Rhee et al.[11] pointed out that KGA may break SCF-PEKS scheme. Therefore, they designed a new SCF-PEKS scheme satisfying the property of Trapdoor Indistinguishability to prevent KGA. In 2013, Zhao et al.[15] proposed an efficient Trapdoor-indistinguishable SCF-PEKS scheme, which has better performance than Rhee et al's scheme. The PEKS mechanisms above tolerate "exact" keyword search instead of supporting spell inconsistent ("common" and "comon") or format error ("PhD" and "Ph.D"), etc. Therefore, Li et al.[10] firstly introduced "Fuzzy Keyword Search" concept into the encrypted model to solve these problems. In 2013, Xu et al.[14] proposed a new PEKS with Fuzzy Keyword Search to resist off-line KGA. Other typical PEKS schemes are also proposed in recent years. Ibraimi et al.[9] proposed PEKS with Delegated Search for detecting encrypted malicious code. Chen et al.[6] formalized Dual-Server PEKS to resist inherent insecurity in 2016. Meanwhile, He et al.[8] proposed a new PEKS scheme which enables users to share contents and subscribe services in mobile social networks. However, these PEKS schemes above specialize in encrypting one keyword only rather than multiple keywords. Baek et al.[1] presented a PEKS scheme to solve multiple keywords search problem but it requires a secure channel to transmit Trapdoor. In 2016, Wang et al.[13] formally defined "Secure Channel

SIN '18, September 10–12, 2018, Cardiff, United Kingdom

ACM ISBN 978-1-4503-6608-3/18/09…$15.00
https://doi.org/10.1145/3264437.3264465

Free Public Key Encryption with Multiple Keywords Search (SCF-MPEKS)" to remove the secure channel. However, SCF-MPEKS may suffer KGA, if the malicious server or receiver release its private key to the public.

This paper formally defines *Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (tSCF-MPEKS)* model and then presents a construction of tSCF-MPEKS and also proves its security under BDH and 1-BDHI assumptions. The proposed scheme has the properties of Ciphertext Indistinguishability and Trapdoor Indistinguishability which is able to resist KGA and CPA.

2 METHODOLOGY

2.1 Bilinear Pairings

Let G_1 and G_T be two cyclic groups (G_1 denotes an additive group and G_T denotes a multiplicative group respectively). g is a generator of G_1 and a large prime number p is the order of G_1. Let x and y be the elements of Z_p. A bilinear pairing can be regarded as a map $e : G_1 \times G_1 \rightarrow G_T$: which has the following properties:
i. Bilinear: $e(xM, yN) = e(M, N)^{xy}$ for all $M, N \in G_1$ and $x, y \in Z_p$.
ii. Computable: $e(M, N) \in G_T$ is computable in a polynomial time algorithm, for any $M, N \in G_1$.
iii. Non-degenerate: $e(M, N) \neq 1$.

2.2 The Bilinear Diffie-Hellman (BDH) assumption[3]

Given P, xP, yP, zP as input (where $x, y, z \in Z_p$), compute $e(P, P)^{xyz} \in G_T$. An algorithm A has an advantage ε in solving BDH assumption in G_1, if $Pr[A(P, xP, yP, zP) = e(P, P)^{xyz}] \geq \varepsilon$. It is considered that BDH assumption holds in G_1 if no t time algorithm has an advantage at least ε in solving BDH assumption in G_1.

2.3 The 1-Bilinear Diffie-Hellman Inversion (1-BDHI) assumption[2]

Given P, xP as input (where $x \in Z_p$), compute $e(P, P)^{\frac{1}{x}}$. An algorithm A has an advantage in solving 1-BDHI assumption in G_1, if $Pr[A(P, xP) = e(P, P)^{\frac{1}{x}}] \geq \varepsilon$. It is considered that 1-BDHI assumption holds in G_1 if no t time algorithm has an advantage at least ε in solving 1-BDHI assumption in G_1.

3 TRAPDOOR-INDISTINGUISHABLE SECURE CHANNEL FREE PUBLIC KEY ENCRYPTION WITH MULTI-KEYWORDS SEARCH

3.1 Generic Model for tSCF-MPEKS

Sender, server and receiver are three participants in tSCF-MPEKS model. More specially, sender is a party creating SCF-MPEKS encryption while receiver is a party creating Trapdoor query. Both the sender and the receiver transmit their encrypted messages to the server. Then, the server runs Test algorithm to check whether two encrypted messages have the same keyword. The details are described as follows:
1. $KeyGen_{Param}(1^n)$: Input 1^n and then produce a common parameter cp.
2. $KeyGen_{Server}(cp)$: Input cp and then produce a public and private key pair (pk_{Ser}, sk_{Ser}) of the server.
3. $KeyGen_{Receiver}(cp)$: Input cp and then produce a public and private key pair (pk_{Rec}, sk_{Rec}) of the receiver.
4. $SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$: Input the server's public key pk_{Ser} and the receiver's public key pk_{Rec}, then generate a searchable encryption S of a keyword-vector $W = (w_1, w_2, ..., w_n)$.
5. $Trapdoor(pk_{Ser}, sk_{Rec}, w)$: Input the server's public key pk_{Ser} and the receiver's private key sk_{Rec}, then generate a trapdoor T_w of a keyword w.
6. $Test(sk_{Ser}, S, T_w)$: Input the server's private key sk_{Ser}, a searchable encryption S=$SCF-MPEKS(pk_{Ser}, pk_{Rec}, W)$ and a trapdoor T_w=$Trapdoor(pk_{Ser}, sk_{Rec}, w)$. If W includes w, output "yes". Otherwise, output "no".

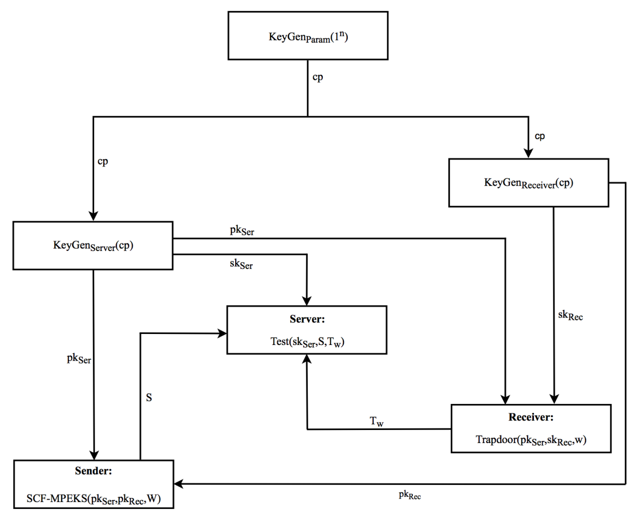

Figure 1: The structure of tSCF-MPEKS model

3.2 Secure Models for tSCF-MPEKS

As discussed in [1, 13], tSCF-MPEKS is IND-CPA and Trapdoor-IND-CPA.

IND-CPA security is that the malicious server could not decide which SCF-MPEKS ciphertext contains which encrypted keyword, if it has not received the Trapdoor containing the given keyword. Besides, if the malicious receiver that has not obtained the server's private key cannot check whether SCF-MPEKS ciphertext and Trapdoor have the same keyword, even if he/she intercepts all Trapdoors for any keyword.

Trapdoor-IND-CPA security is that an outside attacker excluding the server and the receiver cannot differentiate any difference between Trapdoors for two challenge keywords.

The IND-CPA and Trapdoor-IND-CPA for tSCF-MPEKS are formally defined as the following: Let **A** be an attacker whose running time is bounded by t and **E** be a challenger.

Game1: **A** is supposed to be a malicious server.
Setup: The challenger **E** initially runs $KeyGen_{Param}(1^n)$, $KeyGen_{Server}(cp)$ and $KeyGen_{Receiver}(cp)$ to generate a common parameter cp, a public/private key pair (pk_{Ser}, sk_{Ser}) of the server and

a public/private key pair (pk_{Rec}, sk_{Rec}) of the receiver. Then, the attacker **A** receives cp, pk_{Ser}, sk_{Ser} and pk_{Rec}.
Phase 1-1 (Trapdoor queries): Adaptively, the attacker **A** can ask **E** for any trapdoor T_w for any keyword w.
Challenge: **A** sends a target keyword-vector pair (W_0, W_1) on which it wishes to be challenged by **E**, where $W_0 = (w_{01}, ..., w_{0n})$ and $W_1 = (w_{11}, ..., w_{1n})$. Note that W_0 and W_1 cannot be queried in *Phase 1-1*. Once **E** receives the target keyword-vector pair, he/she runs $SCF - MPEKS$ algorithm to generate a searchable encryption S=$SCF - MPEKS(pk_{Ser}, pk_{Rec}, W_\lambda)$, where $\lambda \in \{0, 1\}$. Finally, **E** sends S back to **A** .
Phase 1-2 (Trapdoor queries): **A** can continue to ask **E** for any trapdoor T_w for any keyword w as in *Phase 1-1*, as long as $w \neq w_0, w_1$.
Guess: **A** outputs the guess $\lambda^* \in \{0, 1\}$ and wins ***Game1***, if $\lambda^* = \lambda$.

Game2: **A** is supposed to be a malicious receiver.
Setup: The challenger **E** initially runs $KeyGen_{Param}(1^n)$, $KeyGen_{Server}(cp)$ and $KeyGen_{Receiver}(cp)$ to generate a common parameter cp, a public/private key pair (pk_{Ser}, sk_{Ser}) of the server and a public/private key pair (pk_{Rec}, sk_{Rec}) of the receiver. Then, the attacker **A** receives cp, pk_{Rec}, sk_{Rec} and pk_{Ser}.
Challenge: **A** sends a target keyword-vector pair (W_0, W_1) on which it wishes to be challenged by **E**, where $W_0 = (w_{01}, ..., w_{0n})$ and $W_1 = (w_{11}, ..., w_{1n})$. Notice that $T_{w_{0i}}$ and $T_{w_{1i}}$ cannot be queried in $Test$ algorithm, where $i = 1, ..., n$. Once **E** receives the target keyword-vector pair, he/she runs $SCF-MPEKS$ algorithm to generate a searchable encryption S=$SCF-MPEKS(pk_{Ser}, pk_{Rec}, W_\lambda)$, where $\lambda \in \{0, 1\}$. Finally, **E** sends S back to **A**.
Guess: **A** outputs the guess $\lambda^* \in \{0, 1\}$ and wins ***Game2***, if $\lambda^* = \lambda$.

The advantage of **A** wins ***Game1*** and ***Game2*** is as follows:

$$Adv^{IND-CPA}_{tSCF-MPEKS,A_i}(k) = |Pr[\lambda^* = \lambda] - 1/2|. \quad (i = 1, 2)$$

Therefore, the tSCF-MPEKS model can be regarded as IND-CPA secure only if the $Adv^{IND-CPA}_{tSCF-MPEKS,A_i}(k)$ is negligible.

Game3: **A** is supposed to be an outside attacker excluding the server and the receiver.
Setup: The challenger **E** initially runs $KeyGen_{Param}(1^n)$, $KeyGen_{Server}(cp)$ and $KeyGen_{Receiver}(cp)$ to generate a common parameter cp, a public/private key pair (pk_{Ser}, sk_{Ser}) of the server and a public/private key pair (pk_{Rec}, sk_{Rec}) of the receiver. Then, the attacker **A** receives cp, pk_{Rec}, pk_{Ser} while sk_{Rec}, sk_{Ser} cannot be sent to **A**.
Phase 3-1 (Trapdoor queries): Adaptively, the attacker **A** can ask **E** for any trapdoor T_w for any keyword w.
Challenge: **A** sends a target keyword pair (w_0, w_1) on which it wishes to be challenged by **E**. It should be clear that none of w_0 and w_1 has been queried in *Phase 3-1*. Once **E** receives the target keyword pair, he/she runs $Trapdoor$ algorithm to generate a trapdoor T_w=$Trapdoor(pk_{Ser}, sk_{Rec}, w_\lambda)$, where $\lambda \in \{0, 1\}$. Finally, **E** sends T_w back to **A**.
Phase 3-2 (Trapdoor queries): **A** can continue to ask **E** for any trapdoor T_w for any keyword w as in *Phase 3-1*, as long as $w \neq w_0, w_1$.
Guess: **A** outputs the guess $\lambda^* \in \{0, 1\}$ and wins ***Game3***, if $\lambda^* = \lambda$.

The advantage of **A** wins ***Game3*** is as follows:

$$Adv^{Trap-IND-CPA}_{tSCF-MPEKS,A_3}(k) = |Pr[\lambda^* = \lambda] - 1/2|.$$

Therefore, the tSCF-MPEKS model can be regarded as Trapdoor-IND-CPA secure only if the $Adv^{Trap-IND-CPA}_{tSCF-MPEKS,A_3}(k)$ is negligible.

4 PROPOSED tSCF-MPEKS SCHEME

4.1 The Construction of tSCF-MPEKS

1. $KeyGen_{Param}(k)$: Suppose G_1 is an additive cyclic group and G_T is a multiplicative cyclic group. g is a random generator of G_1 whose order is a prime number $p \geq 2^k$. A bilinear pairing is a map $e : G_1 \times G_1 \to G_T$. Let $H : \{0, 1\}^\circ \to G_1$ and $H^* : G_T \to \{0, 1\}^\bullet$ be two specific hash functions. This algorithm returns a common parameter $cp = \{g, p, G_1, G_T, e, H, H^*\}$.
2. $KeyGen_{Server}(cp)$: The server randomly chooses $a \in Z_p$ and then computes $A = aP$. Besides, the server also chooses $B \in G_1$ uniformly at random. Therefore, the server's public key is $pk_{Ser} = (cp, A, B)$ and the private key is $sk_{Ser} = (cp, a)$.
3. $KeyGen_{Receiver}(cp)$: The receiver randomly chooses $c \in Z_p$ and then computes $C = cP$. Therefore, the receiver's public key is $pk_{Rec} = (cp, C)$ and the private key is $sk_{Rec} = (cp, c)$.
4. $SCF - MPEKS(pk_{Ser}, pk_{Rec}, W)$: The sender randomly chooses $t \in Z_p$ and then computes a searchable encryption $S = (M, N_1, N_2, ..., N_n) = (tA, H^*(D_1), H^*(D_2), ..., H^*(D_n))$, where $D_1 = e(H(w_1), C)^t$, $D_2 = e(H(w_2), C)^t$,..., $D_n = e(H(w_n), C)^t$.
5. $Trapdoor(pk_{Ser}, sk_{Rec}, w^*)$: The receiver randomly chooses $t^* \in Z_p$ and then computes $T_w = (T_1, T_2)$, where $T_1 = cH(w^*) \oplus e(A, B)^{t^*+c}$ and $T_2 = e(A, t^*B)$.
6. $Test(S, T_w, sk_{Ser})$: For $i \in \{1, 2, ..., n\}$, the server initially calculates $T = T_1 \oplus T_2 \bullet e(aB, C) = cH(w^*)$. Then, the server checks if $H^*[e(T, \frac{M}{a})] = N_i$. If so, output "yes"; if not, output "no".

4.2 The Correctness of tSCF-MPEKS

Assuming W is a keyword-vector in $SCF - MPEKS$ algorithm and w^* is a keyword in $Trapdoor$ algorithm respectively. The scheme is correct if W includes w^*. The details are described below: For $i \in \{1, 2, ..., n\}$,
Firstly,

$$\begin{aligned} T &= T_1 \oplus T_2 \bullet e(aB, C) \\ &= cH(w^*) \oplus e(A, B)^{t^*+c} \oplus e(A, t^*B) \bullet e(aB, cP) \\ &= cH(w^*) \oplus e(A, B)^{t^*+c} \oplus e(A, B)^{t^*} \bullet e(A, B)^c \\ &= cH(w^*) \oplus e(A, B)^{t^*+c} \oplus e(A, B)^{t^*+c} \\ &= cH(w^*) \end{aligned}$$

Then,

$$\begin{aligned} H^*[e(T, \frac{M}{a})] &= H^*[e(cH(w^*), \frac{tA}{a})] \\ &= H^*[e(cH(w^*), tP)] \\ &= H^*[e(H(w^*), C)^t] \\ &= N_i \end{aligned}$$

4.3 The Security Analysis of tSCF-MPEKS

THEOREM 4.1. *The tSCF-MPEKS scheme above is IND-CPA secure against CPA in* ***Game1*** *under the random oracle model assuming that BDH assumption is intractable.*

Game1: A *is supposed to be a malicious server.*

PROOF. Suppose that **E** has $(g, p, G_1, G_T, e, xP, yP, zP)$ as an input of BDH assumption whose running time is bounded by T. **E**'s aim is to calculate a BDH key $e(P, P)^{xyz}$ of xP, yP and zP using **A**'s IND-CPA. Besides, suppose that **A** asks for at most h and h^* hash function queries.

Setup Simulation

E firstly sets $C = xP$ and randomly selects $a \in Z_p$ and then calculates $A = aP$. **E** also picks up $B \in G_1$ uniformly at random. Finally, **E** returns $(g, p, G_1, G_T, e, H, H^*)$ as the common parameter *cp*, (*cp*,*A*,*B*) and (*cp*,*a*) as the server's public/private keys and (*cp*,*C*) as the receiver's public key. Besides, **E** chooses two hash functions H and H^* as follows:

- **A** can query a keyword w_i to H function at any time. To respond, **E** searches H_List for a tuple $(w_i, F_i, f_i, \theta_i)$ and the H_List is empty in original. If the sample exists, **A** will receive $H(w_i) = F_i$ as a response. Otherwise, **E** does the following steps:

i. **E** picks up a coin θ_i uniformly at random and then calculates $Pr[\theta_i = 0] = \frac{1}{h+1}$.

ii. **E** selects $f_i \in Z_p$ uniformly at random. If $\theta_i = 0$, **E** will calculate $F_i = yP + f_iP$. If $\theta_i = 1$, **E** will calculate $F_i = f_iP$.

iii. **E** returns F_i as a response to **A** and adds $(w_i, F_i, f_i, \theta_i)$ into H_List.

- **A** can query D_i to H^* function at any time. Then, **E** searches H^*_List for a tuple (D_i, N_i). If the sample exists, **A** will receive N_i as a response. Otherwise, **E** selects $N_i \in \{0, 1\}^d$ uniformly at random and then returns it to **A** and also adds (D_i, N_i) into H^*_List.

Phase 1-1 Simulation (Trapdoor queries)

When **A** issues a query for the trapdoor corresponding to the word w_i. To respond, **E** executes the following steps:

- **E** runs the above algorithm for simulating H function to create a tuple $(w_i, F_i, f_i, \theta_i)$. If $\theta_i = 0$, **E** will stop and output "Suspension". Otherwise, **E** conducts the next step.

- **E** selects $t^* \in Z_p$ and then computes $T_1 = f_iC \oplus e(A, B)^{t^*+x} = f_ixP \oplus e(A, B)^{t^*+x} = xF_i \oplus e(A, B)^{t^*+x} = xH(w_i) \oplus e(A, B)^{t^*+x}$ and $T_2 = e(A, t^*B)$. So, T_w=(T_1,T_2).

Challenge Simulation

A sends W_0=(w_{01},w_{02},...,w_{0n}) and W_1=(w_{11},w_{12},...,w_{1n}) to **E**. Upon receiving the target keyword-vector pair, **E** responds as follows:

- **E** randomly selects $i \in \{1, 2, ..., n\}$.

- **E** runs the above algorithms for simulating H function to obtain two tuples $(w^*_{0i}, F^*_{0i}, f^*_{0i}, \theta^*_{0i})$ and $(w^*_{1i}, F^*_{1i}, f^*_{1i}, \theta^*_{1i})$. If θ^*_{0i} and θ^*_{1i} are equal to 1, **E** will stop and output "Suspension". Otherwise, **E** conducts the next step.

i. **E** runs the above algorithms for simulating H function at $2(n-1)$ times to obtain two vectors of tuples $((w^*_{01}, F^*_{01}, f^*_{01}, \theta^*_{01}), ..., (w^*_{0i-1}, F^*_{0i-1}, f^*_{0i-1}, \theta^*_{0i-1}), (w^*_{0i+1}, F^*_{0i+1}, f^*_{0i+1}, \theta^*_{0i+1}), ..., (w^*_{0n}, F^*_{0n}, f^*_{0n}, \theta^*_{0n}))$ and $((w^*_{11}, F^*_{11}, f^*_{11}, \theta^*_{11}), ..., (w^*_{1i-1}, F^*_{1i-1}, f^*_{1i-1}, \theta^*_{1i-1}), (w^*_{1i+1}, F^*_{1i+1}, f^*_{1i+1}, \theta^*_{1i+1}), ..., (w^*_{1n}, F^*_{1n}, f^*_{1n}, \theta^*_{1n}))$. If θ^*_{0j} and θ^*_{1j} are equal to 0 for all $j = 0, ..., i-1, i+1, ..., n$, **E** will stop and output "Suspension". Otherwise, **E** responds as follows:

– **E** randomly chooses $\beta \in \{0, 1\}^d$.

– **E** randomly chooses $J_j \in \{0, 1\}^d$ and creates a target $SCF-MPEKS$ Ciphertext $S^* = (M^*, N^*_1, N^*_2, ..., N^*_n) = (zA, J_1, J_2, ..., J_n))$

So, $S^* = (M^*, N^*_1, ..., N^*_{i-1}, N^*_{i+1}, ..., N^*_n) = (zA, H^*[e(H(w_{\beta_1}), C)^z], ..., H^*[e(H(w_{\beta_{i-1}}), C)^z], H^*[e(H(w_{\beta_i+1}), C)^z], ..., H^*[e(H(w_{\beta_n}), C)^z])$

Note that $J_j = e(H(w_{\beta_i}), C)^z = e(yP + f_{\beta_i}P, xP)^z = e(yP, xP)^z \bullet e(f_{\beta_i}P, xP)^z = e(P, P)^{xyz} \bullet e(zP, xP)^{f_{\beta i}}$

Note also that $e(f_{\beta_k}P, xP)^z = e(f_{\beta_k}P, C)^z = e(H(w_{\beta_k}), C)^z$

Phase 1-2 Simulation (Trapdoor queries)

A can continue to ask **E** for Trapdoor queries for the keyword w_i. **E** answers **A** as in *Phase 1-1*, as long as $w_i \notin W_0, W_1$.

Guess

A outputs the guess $\beta^* \in \{0, 1\}$. Then, **E** selects d in the list for H^* function and returns $\frac{d_{\beta^*_i}}{e(zP, xP)^{f_{\beta^*_i}}}$ as the guess for BDH key.

Analysis of *Game1*

Three events are customized as follows:

Event1: **E** does not suspend during Phase 1-1 and Phase 1-2 (Trapdoor queries).

Event2: **E** does not suspend during Challenge Simulation.

Event3: **A** does not issue a query for either one of $H^*(e(H(w^*_{0i}), C)^z)$ or $H^*(e(H(w^*_{1i}), C)^z)$.

Claim 1: $Pr[Event1] \geq \frac{1}{e}$

PROOF. Suppose that **A** does not issue the same keyword twice in Trapdoor queries. So, the probability that a Trapdoor query causes **E** for suspension is $\frac{1}{h+1}$. Therefore, due to **A** asks for at most h Trapdoor queries, the probability that **E** does not suspend in all is at least $(1 - \frac{1}{h+1})^h \geq \frac{1}{e}$. □

Claim 2: $Pr[Event2] \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$

PROOF. If $\theta_0 = \theta_1 = 1$, **E** will suspend during Challenge Simulation. So, the probability that **E** does not suspend is $1-(1-\frac{1}{h+1})^2$. Besides, if θ^*_{0j} and θ^*_{1j} are equal to 0 for all $j = 0, ..., i-1, i+1, ..., n$, **E** will also suspend here. Hence, the probability that **E** does not suspend during Challenge Simulation is at least $(1 - \frac{1}{h+1})^{2(n-1)}\{1 - (1 - \frac{1}{h+1})^2\} \geq (\frac{1}{h+1}) \bullet (\frac{h}{h+1})^{2(n-1)}$. □

Claim 3: $Pr[Event3] \geq 2\varepsilon$

PROOF. As discussed in [1], suppose $Hybrid_r$ for $r \in \{1, 2, ..., n\}$ is an *event* that the attacker **A** can successfully guess the keyword of the left part of a "hybrid" $SCF-MPEKS$ Ciphertext formed with r, coordinates from w_β followed by $(n-r)$ coordinates from $w_{1-\beta}$. Consequently, $Pr[Event3] = 2\Sigma^n_{k=1}(Pr[Hybrid_r] - Pr[Hybrid_{r-1}]) = 2(Pr[Hybrid_r] - Pr[Hybrid_0]) = 2\varepsilon$. □

Because the probability that **A** issues a query for either $H^*(e(H(w^*_{0i}), C)^z)$ or $H^*(e(H(w^*_{1i}), C)^z)$ is at least 2ε, the probability that **A** issues a query for $H^*(e(H(w^*_{ji}), C)^z)$ is at least ε. In total, **E**'s success probability ε^* is $(\frac{h}{h+1})^{2(n-1)} \bullet \frac{\varepsilon}{e(h+1)h^*}$.

Theorem 4.2. *The tSCF-MPEKS scheme above is IND-CPA secure against CPA in **Game2** under the random oracle model assuming that 1-BDHI assumption is intractable.*

Game2: ***A*** *is supposed to be a malicious receiver.*

Proof. Suppose that **E** has (g, p, G_1, G_T, e, xP) as an input of 1-BDHI assumption whose running time is bounded by T. **E**'s aim is to calculate a 1-BDHI key $e(P,P)^{\frac{1}{x}}$ of xP using **A**'s IND-CPA. Besides, suppose that **A** asks for at most h and h^* hash function queries.

Setup Simulation
E firstly sets $A = xP$ and $B \in G_1$. **E** also selects $c \in Z_p$ uniformly at random and calculates $C = cP$. Then, **E** returns $(g, p, G_1, G_T, e, H, H^*)$ as the common parameter cp, (cp,A,B) as the server's public key, (cp,C) and (cp,c) as the receiver's public/private keys. Besides, **E** chooses two hash functions H and H^* as follows:
– **A** can query a keyword w_i to H function at any time. To respond, **E** selects $f_i \in Z_p$ uniformly at random and then calculates $F_i = f_i P$ and finally returns F_i as a response to **A**.
– **A** can query D_i to H^* function at any time. Then, **E** searches H^*_List for a tuple (D_i, N_i). If the sample exists, **A** will receive N_i as an answer. Otherwise, **E** selects $N_i \in \{0,1\}^d$ uniformly at random and then returns it to **A** and also adds (D_i, N_i) into H^*_List.
Challenge Simulation
A sends $(W^*_{0i}, F^*_{0i}, f^*_{0i}, \theta^*_{0i})$ and $(W^*_{1i}, F^*_{1i}, f^*_{1i}, \theta^*_{1i})$ to **E**, where $W^*_0 = (w_{01}, w_{02}, ..., w_{0n})$ and $W^*_1 = (w_{11}, w_{12}, ..., w_{1n})$. **E** randomly chooses $J_j \in \{0,1\}^d$ and $\beta \in \{0,1\}^d$. Then, **E** creates a target $SCF-MPEKS$ Ciphertext $S^* = (M^*, N^*_1, N^*_2, ..., N^*_n) = (\psi xP, J_1, J_2, ..., J_n)$.
So, $S^* = (M^*, N^*_1, N^*_2, ..., N^*_n) = (\psi xP, H^*(e(H(w_{\beta_1}), C)^{\psi}), H^*(e(H(w_{\beta_2}), C)^{\psi}), ..., H^*(e(H(w_{\beta_n}), C)^{\psi}))$
Note that $e(H(w_{\beta^*_i}), C)^{\psi}) = e(f_i P, cP)^{\psi} = e(P,P)^{\psi \cdot f_i c}$.
Guess
A outputs the guess $\beta^* \in \{0,1\}$. Then, **E** selects d in the list for H^* function and returns $\psi = \frac{1}{x \cdot f_i c}$ as the guess for 1-BDHI key.

Analysis of *Game2*
Two events are customized as follows:
Event4: **E** does not suspend during Challenge Simulation.
Event5: **A** does not issue a query for either one of $H^*(e(H(w^*_{0i}), C)^{\psi})$ or $H^*(e(H(w^*_{1i}), C)^{\psi})$.

Claim 4: $Pr[Event4] = 1$
Proof. There is no restriction to show that **E** will suspend during Challenge Simulation. Therefore, it is easy to know that $Pr[Event4] = 1$. □

Claim 5: $Pr[\neg Event5] \geq 2\varepsilon$
Proof. When *Event5* happens, it is known that the bit $j \in \{0,1\}$ indicating whether the Ciphertext contains w_{0i} or w_{1i} is independent of **A**'s view. Thus, the probability that **A**'s output j^* satisfying $j = j^*$ is at most $\frac{1}{2}$.
According to Bayes's rule, $Pr[j = j^*] = Pr[j = j^*|Event5]Pr[Event5] + Pr[j = j^*|Event5]Pr[\neg Event5] \leq Pr[j = j^*|Event5]Pr[Even5] + Pr[\neg Event5] = \frac{1}{2} \bullet Pr[Event5] + Pr[\neg Event5] = \frac{1}{2} + \frac{1}{2} \bullet Pr[\neg Event5]$
According to the definition, it is clear that $|Pr[j = j^*] - \frac{1}{2}| \geq \varepsilon$. Then, $\varepsilon \leq Pr[j = j^*] - \frac{1}{2} \leq \frac{1}{2} \bullet Pr[\neg Event5]$. Consequently, $Pr[\neg Event5] \geq 2\varepsilon$. □

Because the probability that **A** issues a query for either $H^*(e(H(w^*_{0i}), C)^{\psi})$ or $H^*(e(H(w^*_{1i}), C)^{\psi})$ is at least 2ε, the probability that **A** issues a query for $H^*(e(H(w^*_{ji}), C)^{\psi})$ is at least ε. Due to **A** asks for at most h^* hash function queries, the probability that **E** selects the correct answer is at least $\frac{1}{h^*}$. In total, **E**'s success probability ε^* is $\frac{\varepsilon}{h^*}$.

Theorem 4.3. *The tSCF-MPEKS scheme above is Trapdoor-IND-CPA secure against CPA in **Game3** under the random oracle model assuming that BDH assumption is intractable.*

Game3: ***A*** *is supposed to be an outside attacker excluding the server and the receiver.*

Proof. Suppose that **E** has $(g, p, G_1, G_T, e, xP, yP, zP)$ as an input of BDH assumption whose running time is bounded by T. **E**'s aim is to calculate a BDH key $e(P,P)^{xyz}$ of xP, yP and zP using **A**'s IND-CPA. Besides, suppose that **A** asks for at most h and h^* hash function queries.

Setup Simulation
E firstly sets $A = xP$, $B = yP$ and $C = zP$ and then returns (cp, A, B) as the server's public key and (cp, C) as the receiver's public key. **E** also randomly chooses two H and H^* hash functions at random.
Phase 3-1 Simulation (Trapdoor queries)
When **A** issues a query for the trapdoor corresponding to the word w_i. To respond, **E** chooses $t^* \in Z_p$ uniformly at random and then computes $T_1 = zH(w_i) \oplus e(yP, xP)^{t^*+z}$ and $T_2 = e(t^* yP, xP)$. So, T_w=(T_1,T_2). Finally, **E** returns T_w to **A**.
Challenge Simulation
A sends (w^*_0, w^*_1) to **E**. **E** creates the challenge Trapdoor as follows: **E** randomly selects a bit $\beta \in \{0,1\}^d$. Therefore, $T_1 = zH(w_{\beta^*}) \oplus e(yP, xP)^{t^*+z} = zH(w_{\beta^*}) \oplus e(P,P)^{xyz} \bullet e(P,P)^{xyt^*}$ and $T_2 = e(t^* yP, xP)$.
Phase 3-2 Simulation (Trapdoor queries)
A can continue to ask **E** for Trapdoor queries for the keyword w_i. **E** answers **A** as in *Phase 3-1*, as long as $w_i \neq w_0, w_1$.
Guess
A outputs the guess $\beta^* \in \{0,1\}$. If $\beta = \beta^*$, **E** outputs "yes". Otherwise, **E** outputs "no".

Analysis of *Game3*
Due to **A** is a malicious outside attacker, he/she cannot distinguish any difference between two Trapdoors even though these two Trapdoors have the same keyword. The reason is that **E** randomly chooses $t^* \in Z_p$ and t^* changes every time leading to $T_1 = cH(w_i) \oplus e(A,B)^{t^*+c}$ changes every time. Even if two Trapdoors have the same keyword, the results are still different because of t^*. Therefore, the key part of Trapdoor Indistinguishability in tSCF-MPEKS is the confidentiality of $e(A,B)^{t^*+c}$.

Suppose that the attacker **A** obtains the value of $e(A,B)^{t^*+c}$, he/she can distinguish whether two Trapdoors have the same keyword.

The reason is that the attacker **A** only calculates one extra XOR operation as $T_1 = cH(w_i) \oplus e(A,B)^{t^*+c} \oplus e(A,B)^{t^*+c} = cH(w_i)$. Therefore, the attack **A** can distinguish that $T_{w_0} = cH(w_0)$ and $T_{w_1} = cH(w_1)$ are equal as long as $w_0 = w_1$. According to Challenge Simulation in ***Game3***, it is known that $e(A,B)^{t^*+c} = e(P,P)^{xyz} \bullet e(P,P)^{xyt^*}$, which satisfies BDH assumption. Consequently, the attacker **A** cannot calculate the value of $e(A,B)^{t^*+c}$ and therefore, he/she cannot compute $T_1 = cH(w_i) \oplus e(A,B)^{t^*+c}$.

5 COMPARISON AND PERFORMANCE

This section presents a comparison of security between the proposed scheme (tSCF-MPEKS) and another two typical schemes (MPEKS[1] and SCF-MPEKS[13]). In addition, the performance of the proposed scheme is also described in this part.

Table 1: Comparison of security assumption and properties

Scheme	CT Ind	Trap Ind	SC	KGA
MPEKS	Satisfied	Not satisfied	Required	Vulnerable to KGA
SCF-MPEKS	Satisfied	Not satisfied	Not required	Vulnerable to KGA
Proposed scheme	Satisfied	Satisfied	Not required	Not vulnerable to KGA

CT Ind, Trap Ind, SC and KGA are the abbreviation of Ciphertext Indistinguishability, Trapdoor Indistinguishability, Secure Channel and Keyword Guessing Attack respectively.

The proposed scheme is simulated using type A pairing in JPBC Library[7]. The conditions of the simulation platform is illustrated in Table 2 and the time cost is shown in Table 3. However, the proposed scheme removes the secure channel so that the trapdoor can be transmitted via the public networks. Also, the proposed scheme satisfies Ciphertext Indistinguishability and Trapdoor Indistinguishability which is able to resist KGA. Overall, the proposed scheme has better performance than Baek et al's MPEKS[1] and Wang et al's SCF-MPEKS[13] schemes.

Table 2: Simulation Platform

OS	macOS Sierra 10.12.5
CPU	2.5 GHz Intel Core i7
Memory	16 GB 1600 MHz DDR3
Hard disk	512GB
Programming language	JAVA

Table 3: Performance by 1000 times computer simulation (n=3)

tSCF-MPEKS	KeyGen_Ser	KeyGen_Rec	SCF-MPEKS	Trapdoor	Test
Average time	0.017s	0.012s	0.088s	0.045s	0.019s

6 CONCLUSION

This paper revisits MPEKS and SCF-MPEKS schemes and then defines the model of *Trapdoor-indistinguishable Secure Channel Free Public Key Encryption with Multi-Keywords Search (tSCF-MPEKS)* and also presents a concrete scheme. The proposed scheme solves multiple keywords search problem and incorporates the advantages of removing secure channel so that it is a practical and cost-saving system. By comparison of security between the tSCF-MPEKS scheme and the others, the proposed scheme satisfying Trapdoor Indistinguishability is much secure and can prevent KGA. In addition, the proposed scheme is efficient and has high performance by 1000 times computer simulation.

REFERENCES

[1] Joonsang Baek, Reihaneh Safavi-Naini, and Willy Susilo. 2008. Public key encryption with keyword search revisited. In *International conference on Computational Science and Its Applications*. Springer, 1249–1259.

[2] Dan Boneh and Xavier Boyen. 2004. Efficient selective-ID secure identity-based encryption without random oracles. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 223–238.

[3] Dan Boneh and Xavier Boyen. 2004. Secure identity based encryption without random oracles. In *Annual International Cryptology Conference*. Springer, 443–459.

[4] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. 2004. Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques*. Springer, 506–522.

[5] Jin Wook Byun, Hyun Suk Rhee, Hyun-A Park, and Dong Hoon Lee. 2006. Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In *Workshop on Secure Data Management*. Springer, 75–83.

[6] Rongmao Chen, Yi Mu, Guomin Yang, Fuchun Guo, and Xiaofen Wang. 2016. Dual-server public-key encryption with keyword search for secure cloud storage. *IEEE transactions on information forensics and security* 11, 4 (2016), 789–798.

[7] Angelo De Caro and Vincenzo Iovino. 2011. jPBC: Java pairing based cryptography. In *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*. Kerkyra, Corfu, Greece, June 28 - July 1, 850–855.

[8] Zaobo He, Zhipeng Cai, Qilong Han, Weitian Tong, Limin Sun, and Yingshu Li. 2016. An energy efficient privacy-preserving content sharing scheme in mobile social networks. *Personal and Ubiquitous Computing* 20, 5 (2016), 833–846.

[9] Luan Ibraimi, Svetla Nikova, Pieter Hartel, and Willem Jonker. 2011. Public-key encryption with delegated search. In *International Conference on Applied Cryptography and Network Security*. Springer, 532–549.

[10] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. 2010. Fuzzy keyword search over encrypted data in cloud computing. In *Infocom, 2010 proceedings ieee*. IEEE, 1–5.

[11] Hyun Sook Rhee, Jong Hwan Park, Willy Susilo, and Dong Hoon Lee. 2010. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *Journal of Systems and Software* 83, 5 (2010), 763–771.

[12] Qiang Tang and Liqun Chen. 2009. Public-key encryption with registered keyword search. In *European Public Key Infrastructure Workshop*. Springer, 163–178.

[13] Tingting Wang, Man Ho Au, and Wei Wu. 2016. An efficient secure channel free searchable encryption scheme with multiple keywords. In *International Conference on Network and System Security*. Springer, 251–265.

[14] Peng Xu, Hai Jin, Qianhong Wu, and Wei Wang. 2013. Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack. *IEEE Transactions on computers* 62, 11 (2013), 2266–2277.

[15] Yuanjie Zhao, Xiaofeng Chen, Hua Ma, Qiang Tang, and Hui Zhu. 2012. A New Trapdoor-indistinguishable Public Key Encryption with Keyword Search. *JoWUA* 3, 1/2 (2012), 72–81.