# Improved Secure Server-Designated Public Key Encryption with Keyword Search

Nitish Andola
*Department of IT*
*IIIT-Allahabad*
U.P, India
andola.nitish18@gmail.com

Sourabh Prakash
*Department of IT*
*IIIT-Allahabad*
U.P, India
sprakash13@gmail.com

S. Venkatesan
*Department of IT*
*IIIT-Allahabad*
U.P, India
venkat@iiita.ac.in

Shekhar Verma
*Department of IT*
*IIIT-Allahabad*
U.P, India
sverma@iiita.ac.in

***Abstract*—Cloud storage service serves the need of data sharing between multiple parties. Cloud storage services allow the data owners to outsource their encrypted records to store over cloud storage servers and authorizes multiple users to access these files. Searching over encrypted data at cloud environment was made possible by various searchable schemes. Many of such schemes has its own limitations and security concerns. Existing schemes suffers from ON-KGA in which a malicious launches a keyword guessing attack to find the keyword used for generating the trapdoor. In this paper, we introduce a scheme which is immune to online Keyword guessing attacks(ON-KGA) in an environment where a server can be malicious. We propose a nonce based scheme in which a nonce is added with the keyword, thereby removing the weakness of ON-KGA, we also enhanced the existing security model for TD-IND.**

***Index Terms*—Bilinear Pairing, Cloud Security, Keyword Search, Searchable Encryption**

## I. Introduction

In recent years, a great demand of cloud architecture for storing the users data has being noticed. User use their laptops, smartphones to store their private data over the cloud. In Cloud environment there is a problem of data disclosure to an adversary or server hosting the data. Encryption does not solve the problem of Privacy Information Retrieval(PIR) [1] as it does not allow part download of encrypted data. This can be achieved through Symmetric Encryption with Keyword Search(SEKS) [2], in which a user uploads the encrypted data alongwith a list of searchable keywords for different part of data. A user can download a part of encrypted data which is attached to a specific searchable keyword. But the user who is owner of the data cannot share data with other users. In public key encryption with keyword search(PEKS) [3] a set of searchable keywords are provided, where the server after validating the keyword with the help of trapdoor set by the receiver, sends the message to the receiver.

**Related Work**: First PEKS [3] introduced the security model for searchable ciphertext indistinguishability, in which a secure channel was required.Later, its found that in absence of a secure channel, any outsider can link the trapdoor with the matching PEKS ciphertext [4]. So they introduced another security model for secure channel free SC-IND. However, their security model could not meet the requirements in practical scenerio, as the abilities of the attacker are bounded as for launching a attack, private key is required with the help of a malicious server or a receiver. In [5] it has been listed as an open problem to construct PEKS scheme which provides security against Keyword Guessing Attack. Afterwards [6] mentioned that trapdoor privacy in the PEKS scheme is almost impossible to achieve in a public-key scenerio. Then [7] also defined a security model for trapdoor indistinguishability(TD-IND), and introduced a scheme known as designated server(dPEKS) scheme to prevent OF-KGA. However, [8] discovered that dPEKS cannot withstand the online keyword guessing attack(ON-KGA). Recently, [9] defined a security model for original ciphertext indistinguishability(OC-IND), and proposed a new framework, called secure server-designated public key encryption with keyword search(SPEKS), to prevent ON-KGA, but this framework is also insecure in case when a malicious server launches a keyword guessing attack to find the keyword used for generating the trapdoor, which was not considered in the related work.

**Our Contribution:** The current schemes do not provide proper security model for trapdoor indistinguishablity (TD-IND), SPEKS provides TD-IND only with respect to outsider, the malicious server can launch keyword guessing attack(KGA) to find the keyword forming the trapdoor. No solution was provided in [10] to remove this weakness. In the proposed scheme, the malicious server can recover the keyword by launching the KGA to the trapdoor. Our contribution is as follows:

1) We propose a solution to overcome the weakness in SPEKS, so that the server is unable to guess the keyword forming the trapdoor.
2) We enhance the existing security model for trapdoor indistinguishability to remove this weakness. Our enhanced security model guarantees that an adversary whether server or outsider cannot guess the keyword forming the trapdoor.
3) Lastly, we propose a new framework for removing this weakness.

## II. SPEKS and SPEKS Weakness

Here we first discuss the SPEKS scheme followed by weakness in this scheme.

978-1-5386-1866-0/17/$31.00 

### A. Definition

The probabilistic polynomial time algorithms required for SPEKS scheme are mentioned below. Here Bob is treated as sender while Alice as receiver.

1) GlobalSetup($k$): The input to this algorithm is a security parameter k, and the output is System Parameter $SP$.
2) dKeyGen($SP$): The input to this algorithm is SP, the algorithm runs TE.KeyGen to generate $R_{pub}, R_{priv}$ and the ouptput is $S_{pub} = R_{pub}$ and $S_{priv} = R_{priv}$.
3) rKeyGen($SP$): The input to this algorithm is SP, the algorithm runs IBE.Setup to generate $A_{pub}, A_{priv}$ and the ouptput is $P^y = P_{pub}$ and $y = P_{priv}$
4) dPEKS($SP$, $W_i, A_{pub}, S_{pub}$): Here Bob with the help of Alice's public key and keyword $W \in \mathcal{W}$ generates a searchable ciphertext $C$. Now Bob sends $n$ such searchable ciphertexts along with a encyrpted message $\bar{M}$, where $\bar{M}$ is the traditional encryption of real message $M \in \mathcal{M}$ with the help of Alice's public key.
5) Trapdoor($SP, W^{'}, A_{priv}, S_{pub}$): Here Alice with the help of its own private key and a keyword $W^{'} \in \mathcal{W}$ generates a $trapdoor\ T$ and sends it to the server.
6) Test($SP$, $C$, $T$, $S_{priv}$): Here the input to the algorithm is server's private key $S_{priv}$ and the output is 1 if $W^{'} = W$ else 0.
7) Encrypt($SP, A_{pub}, \bar{M}$): The input to the algorithm is $\bar{M}$, where, $\bar{M} = E_{A_{pub}}[M]$ public key of Alice $A_{pub}$ and the output is a ciphertext $\bar{C}$ of the message $\bar{M}$ with the help of Traditional Encryption.
8) Decrypt($SP, A_{priv}, \bar{C}$): The input to the algorithm is $\bar{C}$, private key of Alice $A_{priv}$ and output is the decrypted plaintext $\bar{M}$.

### B. Weakness in SPEKS

Here, we discuss two weaknesses in SPEKS as mentioned in [10]

1) Given a trapdoor a malicious server can find the keyword used for generating the trapdoor, ie the trapdoor is distinguishable when the server is malicious, which is not considered in the related work.
2) The security model for TD-IND defined in SPEKS is incomplete, the trapdoor is infact is distinguishable.

## III. Proposed New Scheme

In the proposed scheme, the entities involved are sender, server and the receiver.

The main idea is as follows:

1) The receiver selects a set of senders who can generate a ciphertext for receiver, and secretly sends the nonce generated for a specific set of senders by the receiver. This nonce will be transferred only once ie. the receiver will not have to send this for every communication between sender and receiver.
2) Whenever a valid sender needs to send message with searchable keyword, he modified the keyword with the nonce provided by the receiver and forms ciphertext using public key of receiver. He sends the message alongwith the ciphertext to server.
3) The receiver forms a trapdoor with a specific keyword and the same nonce, and finally encrypts the trapdoor with servers public key and provides it to server.
4) With the help of trapdoor stored at the server, the server checks whether the sender's keyword used to form the ciphertext is same as the keyword used to set the trapdoor, if yes, he sends the message (corresponding to the valid ciphertext) to the receiver.

Compared to SPEKS, we add a new algorithm fGen to generate the nonce at the receiver, it then secretly sends the nonce to specific set of senders.

### A. Security Models

Before presenting the security models, we show following four oracles, Trapdoor, Test, Decrypt and dPEKS. We assume $\mathcal{A}$ to be an adversary.

1) Trapdoor: $\mathcal{A}$ sends a keyword $W^{'}$ to given oracle. A trapdoor $T$ is generated and sent back in return.
2) Test: A searchable ciphertext $C$ and a trapdoor $T$ is provided to the oracle by $\mathcal{A}$. If $W = W^{'}$, it outputs 1 else 0.
3) Decrypt: A ciphertext $\bar{C}$ is given to oracle by $\mathcal{A}$. Encrypted Message $\bar{M}$ is sent back in return.
4) dPEKS: $\mathcal{A}$ sends a keyword $W^{'} \in \mathcal{W}$ to given oracle. It returns a searchable ciphertext $C$ for this keyword $W^{'}$.

### B. Security Models for SC-IND

In these security models, the adversary $\mathcal{A}$ acts either as a malicious server or a receiver. SC-IND guarantees that the searchable ciphertexts formed by the two keywords ${W_0}^*$ and ${W_1}^*$ are indistinguishable for an adversary. The security model for SC-IND is the following games between $\mathcal{A}_1$(or $\mathcal{A}_2$) and a challenger $\mathcal{C}$.

1) Game 1 (SC-IND1): Let $\mathcal{A}_1$ be malicious server. The game is performed as follows.
   a) Setup: Here $\mathcal{A}_1$ generates its public and private keys as $pk_S, sk_S$ respectively and sends $pk_S$ to $\mathcal{C}$. $\mathcal{C}$ also generates its public and private keys as $pk_R, sk_R$ respectively and sends $pk_R$ to $\mathcal{A}_1$.
   b) Phase 1: $\mathcal{A}_1$ can send a keyword $W \in \mathcal{W}$ to trapdoor oracle and the dPEKS oracle. In response trapdoor oracle returns a trapdoor $T$ and dPEKS oracle returns a searchable ciphertext $C$.
   c) Challenge: $\mathcal{A}_1$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{C}$. But condition here is that ${W_0}^*, {W_1}^*$ should not have been queried to trapdoor oracle.In return, $\mathcal{C}$ forms a searchable ciphertext $C^*$ with for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen.
   d) Phase 2: $\mathcal{A}_1$ is again allowed to query to trapdoor and dPEKS oracles with a condition that the keywords ${W_0}^*, {W_1}^*$ should not be queried.
   e) Guess: Finally, $\mathcal{A}_1$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_1$ succeeds if $b = b^{'}$.

2) Game 2 (SC-IND2): Let $\mathcal{A}_2$ be the receiver. This game is performed as follows.
   a) Setup: Here $\mathcal{A}_2$ generates its public and private keys as $pk_R, sk_R$ respectively and sends $pk_R$ to $\mathcal{C}$. $\mathcal{C}$ also generates its public and private keys as $pk_S, sk_S$ respectively and sends $pk_S$ to $\mathcal{A}_2$. Nonce is sent to $\mathcal{C}$ by $\mathcal{A}_2$ .
   b) Phase1: Here $\mathcal{A}_2$ submits $(C, T)$ to test oracle for a keyword $W \in \mathcal{W}$. In return oracle output 0 or 1.
   c) Challenge: $\mathcal{A}_2$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{C}$. But condition here is that ${W_0}^*, {W_1}^*$ should not have been queried to test oracle. In return, $\mathcal{C}$ forms a searchable ciphertext $C^*$ with for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen.
   d) Phase 2: $\mathcal{A}_2$ is again allowed to query to test oracle with a condition that the trapdoor $T$ and searchable ciphertext $C$ for the keywords ${W_0}^*, {W_1}^*$ should not be the input of Test oracle.
   e) Guess: Finally, $\mathcal{A}_2$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_2$ succeeds if $b = b^{'}$.

The advantage of $\mathcal{A}_1$ or $\mathcal{A}_2$ to break our scheme is ${Adv_{\mathcal{A}_1 or \mathcal{A}_2}}^{SC-IND}(k) = \left| Pr[b = b^{'}] - \frac{1}{2} \right|$.

DEFINITION 1 With respect to adaptive chosen keyword attack, our scheme is secure if ${Adv_{\mathcal{A}_1 or \mathcal{A}_2}}^{SC-IND}(k)$ is negligible for any $PPT$ adversary $A_i (i = 1, 2)$.

### C. Security for OC-IND

OC-IND guarantees that the original ciphertexts formed by the two messages ${\bar{M_0}}^*, {\bar{M_1}}^*$ are indistinguishable for an adversary. The security model for OC-IND is the following games between $\mathcal{A}_3$ and a challenger $\mathcal{C}$.

Game (OC-IND): Let $\mathcal{A}_3$ be an outsider. This game is performed as follows.

1) Setup: $\mathcal{C}$ runs GlobalSetup($k$), KeyGen1($SP$) and KeyGen2($SP$), and outputs $(pk_S, sk_S)$ and $(pk_R, sk_R)$. It will keep $(sk_S, sk_R)$ secret and $(pk_S, pk_R)$ to $\mathcal{A}_3$.
2) Phase 1: $\mathcal{A}_3$ can send a ciphertext $\bar{C}$ to Decrypt oracle. In response it returns $\bar{M}$.
3) Challenge : $\mathcal{A}_3$ sends two messages ${\bar{M_0}}^*, {\bar{M_1}}^*$ for challenge to $\mathcal{C}$. In return, $\mathcal{C}$ forms a ciphertext $C^*$ with for the message ${\bar{M_b}}^*$ where $b \in \{0, 1\}$ is randomly chosen.
4) Phase 2: $\mathcal{A}_3$ is again allowed to query Decrypt with the condition that $\bar{C} \neq \bar{C^*}$.
5) Guess: Finally, $\mathcal{A}_3$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_3$ succeeds if $b = b^{'}$.

The advantage of $\mathcal{A}_3$ to break our scheme is ${Adv_{\mathcal{A}_3}}^{OC-IND}(k) = |Pr[b = b^{'}] - \frac{1}{2}|$. DEFINITION 2. With respect to adaptive chosen keyword attack, our scheme is secure if ${Adv_{\mathcal{A}_3}}^{OC-IND}(k)$ is negligible for any $PPT$ adversary $\mathcal{A}_3$.

### D. Security for TD-IND

In these security models, the adversary $\mathcal{A}$ acts as outsider or a malicious server. TD-IND guarantees that the trapdoors formed by the two keywords ${W_0}^*$ and ${W_1}^*$ are indistinguishable for an adversary. The security model for TD-IND is the following games between $\mathcal{A}_4$(or $\mathcal{A}_5$) and a challenger $\mathcal{C}$.

1) Game1 (TD-IND1): Let $\mathcal{A}_4$ be an outsider.The game is performed as follows.
   a) Setup: $\mathcal{C}$ runs GlobalSetup($k$), KeyGen1($SP$) and KeyGen2($SP$), and then gets $(pk_S, sk_S)$ and $(pk_R, sk_R)$. It will keep $(sk_S, sk_R)$ secret and gives $pk_S, pk_R$ to $\mathcal{A}_4$.
   b) Phase 1: $\mathcal{A}_4$ is allowed to access the trapdoor oracle, dPEKS oracle as well as Test oracle for a keyword $W \in \mathcal{W}$.
   c) Challenge: $\mathcal{A}_4$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{C}$. But condition here is that ${W_0}^*, {W_1}^*$ should not have been queried to trapdoor oracle. In return, $\mathcal{C}$ forms a trapdoor $T^*$ with for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen.
   d) Phase 2: $\mathcal{A}_4$ is again allowed to query trapdoor oracle, dPEKS oracle and Test oracle with the condition that ${W_0}^*, {W_1}^*$ are not allowed to access these oracles.
   e) Finally, $\mathcal{A}_4$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_4$ succeeds if $b = b^{'}$.
2) Game2 (TD-IND2): Let $\mathcal{A}_5$ be the malicious server.This game is performed as follows.
   a) Setup: Here $\mathcal{A}_5$ generates its public and private keys as $pk_S, sk_S$ respectively and sends $pk_S$ to $\mathcal{C}$. $\mathcal{C}$ also generates its public and private keys as $(pk_R, sk_R)$ respectively and sends $pk_R$ to $\mathcal{A}_5$.
   b) Phase 1: $\mathcal{A}_5$ can send a keyword $W \in \mathcal{W}$ to trapdoor oracle and the dPEKS oracle. In response trapdoor oracle returns a trapdoor $T$ and dPEKS oracle returns a searchable ciphertext $C$.
   c) Challenge: $\mathcal{A}_5$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{C}$. But condition here is that ${W_0}^*, {W_1}^*$ should not have been queried to trapdoor oracle.In return, $\mathcal{C}$ forms a trapdoor $T^*$ with for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen.
   d) Phase 2: $\mathcal{A}_5$ is again allowed to query to trapdoor and dPEKS oracles with a condition that the keywords ${W_0}^*, {W_1}^*$ should not be queried.
   e) Guess: Finally, $\mathcal{A}_5$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_5$ succeeds if $b = b^{'}$.

The advantage of $\mathcal{A}_4$ or $\mathcal{A}_5$ to break our scheme is ${Adv_{\mathcal{A}_4 or \mathcal{A}_5}}^{SC-IND}(k) = \left| Pr[b = b^{'}] - \frac{1}{2} \right|$.

DEFINITION 3 With respect to adaptive chosen keyword attack, our scheme is secure if ${Adv_{\mathcal{A}_4 or \mathcal{A}_5}}^{SC-IND}(k)$ is negligible for any $PPT$ adversary $\mathcal{A}_i (i = 4, 5)$.

## IV. The Concrete Scheme

### A. The Construction

The concrete scheme is as follows:

1) Global setup
   This algorithm first picks a large prime number $p$ randomly, and then generates:

a) Two groups $G_1, G_2$ of prime order p.
b) A bilinear map $e : G_1 \times G_2$
c) Three hash functions as $H : \{0,1\}^* \rightarrow G_1, H_1 : \{0,1\}^* \rightarrow G_1$ and $H_2 : G_2 \rightarrow \{0,1\}^{\lambda}$, where $\lambda$ is the security parameter.
d) A random generator $P \in G_1$. Finally the algorithm outputs the System Parameter $SP$, where $\{e, G_1, G_2, H, H_1, H_2, P\} \in SP$

2) KeyGen1$(SP)$
The algorithm takes input $SP$ as input. It runs IBE.Setup to output the master key pair $(sk_S, pk_S)$. Finally it outputs server's public and private keys as $P^x = pk_S$ and $x = sk_S$.
3) KeyGen2$(SP)$
The algorithm takes input $SP$ as input. It runs IBE.Setup to output the master key pair $(sk_R, pk_R)$. Finally, it outputs the receiver's public and private keys as $P^y = pk_R$ and $y = sk_R$.
4) fGen$(\xi)$
A random nonce is chosen from Nonce space $\xi$.
5) dPEKS$(SP, pk_R, pk_S, W, nonce)$
This algorithm proceeds as follows:

a) Input of first phase is a keyword $W \in \mathcal{W}, nonce$, and the output is computed as $\bar{W} = W||nonce$ .
b) Now the input to second phase is a random number $r \in Z_p$, $\bar{W}$, receiver's and sender's public keys, $pk_R$ and $pk_S$ respectively, output is a ciphertext $C = (c_1, c_2)$, where,
i) $c_1 = {pk_R}^r$
ii) $c_2 = H_2(e(pk_S, H_1(\bar{W})^r))$
$\hat{C}$ = $\{c_1, c_2\}$

Sender outputs the searchable ciphertext $C$=TE.Encrypt$(SP, \hat{C}, pk_S)$. It then sends $\bar{M}$ with $n$ searchable ciphertexts $(C_1, ..., C_n)$ to the server, where $\bar{M}$ = TE.Encrypt$(SP, M, P_{pub})$ for the real message $M$.
6) Trapdoor$(pk_S, nonce, pk_S, sk_R, W^{'})$
This algorithm proceeds in following steps:

a) The input is a keyword $W^{'} \in W$, along with $nonce$, and the output is computed as $\bar{W}^{'} = W^{'}||nonce$
b) The receiver generates the trapdoor as

$$\hat{T} = (T_1, T_2) = (P^{r^{'}}, H_1(\bar{W}^{'})^{\frac{1}{y}}.H({pk_S}^{r^{'}})),$$

where $r^{'} \in Z_p$ is randomly chosen.

Finally the trapdoor $T$ = TE.Encrypt$(SP, \hat{T}, pk_S)$ is sent to server.
7) Test$(SP, C, T, sk_S)$
The testing algorithm runs in 2 steps:

a) The server recovers $\hat{C}$ = TE.Decrypt$(SP, C, sk_S)$ and $\hat{T}$ = TE.Decrypt$(SP, T, sk_S)$
b) The input is a trapdoor $T, T_1$, private key of server $sk_S$ and the output is computed as

$$T_3 = T_2/H({T_1}^{sk_S})).$$

c) The server checks whether

$$H_2(e(c_1, {T_3}^{sk_S})) = c_2$$

8) Encrypt$(SP, pk_R, \bar{M})$
This algorithm takes $SP$, the receiver $R'$s public key $pk_R$ and a message $M$ as inputs, outputs $\bar{C}$ = TE.Encrypt$(SP, \bar{M}, pk_R)$, here $\bar{M}$ = TE.Encrypt$(SP, M, pk_R)$.
9) Decrypt$(SP, sk_R, \bar{C})$
This algorithm takes $SP$, the receiver $R$'s private key $sk_R$ and a ciphertext $\bar{C}$ as inputs, gets $M$ by computing $M$ = TE.Decrypt$(SP, sk_R,$ TE.Decrypt$(SP, sk_R, \bar{C}))$.

### *B. Security analysis*

We provide the security models for our proposal. THEOREM 1. Our scheme gives security for TD-IND, SC-IND as well as OC-IND. We provide Lemmas 1-5 to support the above theorem. In the proof mentioned below, we represent adversary of our scheme as $\mathcal{A}_i$ for Game i(i = 1, ..., 5), $\mathcal{B}$ for IBE, $\mathcal{F}$ for TE $\mathcal{C}$ as the challenger.

LEMMA 1. We assume that $\mathcal{A}_1$ breaks SC-IND with advantage ${Adv_{\mathcal{A}_1}}^{SC-IND}(k)$, then their exists another adversary $\mathcal{B}$ which breaks the ANON-sID-CPA of IBE.

Proof. Here the adversary $\mathcal{B}$ will simulate the adversary $\mathcal{A}_1$ for breaking the ANON-sID-CPA of IBE.

1) Setup: $\mathcal{A}_1$ generates its public and private keys as $pk_S, sk_S$ respectively and sends $pk_S$ to $\mathcal{C}$. $\mathcal{C}$ also generates its public and private keys as $pk_R, sk_R$ respectively and sends $pk_R$ to $\mathcal{A}_1$.
2) Phase1: $\mathcal{A}_1$ is allowed to send a trapdoor query. Whenever $\mathcal{A}_1$ sends a trapdoor query for keyword $W \in \mathcal{W}$ to $\mathcal{B}$, $\mathcal{B}$ after receiving W sends ID = W to IBE. Extract and recovers $sk_U$ of ID. Now $\mathcal{B}$ generates trapdoor $T$ = TE.Encrypt$(SP, pk_S, sk_U)$ and sends it to $\mathcal{A}_1$.
3) Challenge: $\mathcal{A}_1$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{B}$. But condition here is that ${W_0}^*, {W_1}^*$ should not have been sent as Trapdoor query. Now $\mathcal{B}$ sends ${ID_0}^* = {W_0}^*, {ID_1}^* = {W_1}^*$ for challenge to $\mathcal{C}$. In return, $\mathcal{C}$ forms a searchable ciphertext $C^*$ with for the keyword ${W_b}^*$ where $b \in \{0,1\}$ is randomly chosen and sends $C^*$ to $\mathcal{B}$. Again, $\mathcal{B}$ sends this $C^*$ to $\mathcal{A}_1$.
4) Phase 2: $\mathcal{A}_1$ is again allowed to send trapdoor query with a condition that the keywords ${W_0}^*, {W_1}^*$ should not be queried.
5) Guess: Finally, $\mathcal{A}_1$ outputs its result as $b^{'} \in \{0,1\}$. $\mathcal{A}_1$ succeeds if $b = b^{'}$. and hence $\mathcal{B}$ also succeeds Game ANON-sID-CPA.

We assume that $\mathcal{A}_1$ succeeds the game in polynomial time. But $\mathcal{B}$ simulates $\mathcal{A}_1$ and therefore $\mathcal{B}$ breaks the ANON-sID-CPA in polynomial time. Hence, ${Adv_{\mathcal{A}_1}}^{SC-IND}(k) \leq {Adv_{B.IBE}}^{ANON-sID-CPA}(k)$

and that ${Adv_{\mathcal{A}_1}}^{SC-IND}(k)$ is equivalent to ${Adv_{\mathcal{B}.IBE}}^{ANON-sID-CPA}(k)$.
We conclude that the proof of the lemma is completed.

LEMMA 2. We assume that $\mathcal{A}_2$ breaks SC-IND with advantage ${Adv_{\mathcal{A}_2}}^{SC-IND}(k)$, then their exists another adversary $\mathcal{B}$ which breaks the IND-CCA of TE.
Proof. The adversary $\mathcal{F}$ will simulate the adversary $\mathcal{A}_2$ for breaking the IND-CCA of TE.

1) Setup: Here $\mathcal{A}_2$ generates its public and private keys as $pk_R, sk_R$ respectively and sends $pk_R$ to $\mathcal{C}$. $\mathcal{C}$ also generates its public and private keys as $pk_S, sk_S$ respectively and sends $pk_S$ to $\mathcal{A}_2$. Nonce is sent to $\mathcal{C}$ by $\mathcal{A}_2$.
2) Phase 1: Here $\mathcal{A}_2$ is allowed to send as Test query for $(C, T)$ as input to $\mathcal{F}$, where $W \in \mathcal{W}$. Now $\mathcal{F}$ sends $C$ and $T$ to TE.Decrypt$(SP, C, sk_S)$ and TE.Decrypt$(SP, T, sk_S)$ respectively and recovers $\hat{C}$ and $\hat{T}$, further performing IBE.Decrypt$(SP, \hat{C}, \hat{T})$ and outputs 1 if equation holds else 0.
3) Challenge: $\mathcal{A}_2$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{F}$, but the condition here is that $C$ and $T^*$ for ${W_0}^*, {W_1}^*$ must not have been queried in phase 1. $\mathcal{F}$ sends${W_0}^*, {W_1}^*$ as a challenge to $\mathcal{C}$. In return, $\mathcal{C}$ forms a searchable ciphertext $C^*$ with for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen and sends $C^*$ to $\mathcal{F}$. Again, $\mathcal{F}$ sends this $C^*$ to $\mathcal{A}_2$.
4) Phase 2: $\mathcal{A}_2$ is allowed to send Test queries for $W \in \mathcal{W}$ with the condition that the trapdoor $T^*$ should not be formed by ${W_0}^*, {W_1}^*$.
5) Guess: Finally, $\mathcal{A}_2$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_2$ succeeds if $b = b^{'}$. and hence $\mathcal{B}$ also succeeds Game IND-CCA.

We assume that $\mathcal{A}_2$ succeeds the game in polynomial time. But $\mathcal{F}$ simulates $\mathcal{A}_2$ and therefore $\mathcal{F}$ breaks the IND-CCA in polynomial time. Hence, ${Adv_{\mathcal{A}_2}}^{SC-IND}(k) \leq {Adv_{\mathcal{F}.TE}}^{IND-CCA}(k)$ is equivalent to ${Adv_{\mathcal{F}.TE}}^{IND-CCA}(k)$.
We conclude that the proof of the lemma is completed.

LEMMA 3. We assume that $\mathcal{A}_3$ breaks OC-IND with advantage ${Adv_{\mathcal{A}_3}}^{OC-IND}(k)$, then their exists another adversary $\mathcal{F}$ which breaks the IND-CCA of TE.

Proof. The adversary $\mathcal{F}$ will simulate the adversary $\mathcal{A}_3$ for breaking the IND-CCA of TE.

1) Setup: $\mathcal{C}$ runs the algorithm GlobalSetup(k) as well as KeyGen1(SP) and sends $pk_R$ to $\mathcal{A}_3$.
2) Phase 1: $\mathcal{A}_3$ is allowed to send decryption queries to $\mathcal{F}$. After receiving $\bar{C}$ $\mathcal{F}$ sends it to Decrypt, and returns $\bar{M}$ to $\mathcal{A}_3$.
3) Challenge: $\mathcal{A}_3$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{F}$. Now $\mathcal{F}$ sends ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{C}$.In return, $\mathcal{C}$ forms a ciphertext $\bar{C}^*$ for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen and sends $\bar{C}^*$ to $\mathcal{F}$. Again, $\mathcal{F}$ sends this $\bar{C}^*$ to $\mathcal{A}_3$.
4) Phase 2: $\mathcal{A}_3$ is allowed to send decryption queries to $\mathcal{F}$ with the condition that $C \neq C^*$.
5) Guess: Finally, $\mathcal{A}_3$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_3$ succeeds if $b = b^{'}$. and hence $\mathcal{F}$ also succeeds Game IND-CCA.

Here we assume that $\mathcal{A}_2$ succeeds the game in polynomial time. But $\mathcal{F}$ simulates $\mathcal{A}_2$ and therefore $\mathcal{F}$ breaks the IND-CCA in polynomial time. Hence, ${Adv_{\mathcal{A}_3}}^{TD-IND}(k) \leq {Adv_{\mathcal{F}.TE}}^{IND-CCA}(k)$ and that ${Adv_{\mathcal{A}_3}}^{TD-IND}(k)$ is equivalent to ${Adv_{\mathcal{F}.TE}}^{IND-CCA}(k)$.
We conclude that the proof of the lemma is completed.

LEMMA 4 Here we assume that $\mathcal{A}_4$ breaks TD-IND with advantage ${Adv_{\mathcal{A}_4}}^{TD-IND}(k)$, then their exists another adversary $\mathcal{F}$ which breaks the IND-CCA of TE.

Proof. Here the adversary $\mathcal{F}$ will simulate the adversary $\mathcal{A}_4$ for breaking the IND-CCA of TE.

1) Setup: $C$ runs GlobalSetup(k), KeyGen1(SP), and KeyGen2(SP), and then gets$(pk_R, sk_R)$ and $pk_S$. It will keep $sk_R$ secret and send $pk_S$. It will keep $sk_R$ secret and send $pk_S, pk_R$ to $\mathcal{A}_4$ and $pk_S, pk_R$ to $\mathcal{F}$.
2) Phase 1: Here $\mathcal{A}_4$ is allowed to send a trapdoor query to $\mathcal{F}$. Tn return $\mathcal{F}$ generates a trapdoor $T$ = TE.Encrypt$(SP, pk_S, IBE.Extract(SP, pk_R, W))$ and returns it to $\mathcal{A}_4$. Also, $\mathcal{A}_4$ is allowed to send as Test query for $(C, T)$ as input to $\mathcal{F}$. Now $\mathcal{F}$ sends $C$ and $T$ to TE.Decrypt$(SP, C, sk_S)$ and TE.Decrypt$(SP, T, sk_S)$ respectively and recovers $\hat{C}$ and $\hat{T}$, further performing IBE.Decrypt$(SP, \hat{C}, \hat{T})$ and outputs 1 if equation holds else 0.
3) Challenge: $\mathcal{A}_4$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{F}$, but the condition here is that $C$ and $T^*$ for ${W_0}^*, {W_1}^*$ must not have been queried in phase 1. Now $\mathcal{F}$ sends ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{C}$. In return, $\mathcal{C}$ forms a trapdoor $T^*$ for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen and sends to $\mathcal{F}$. Again, $\mathcal{F}$ sends this $T^*$ to $\mathcal{A}_4$.
4) Phase 2: $\mathcal{A}_4$ is allowed to send trapdoor queries as in phase 1 but with a condition that keyword chosen should not be ${W_0}^*, {W_1}^*$.
5) Guess: Finally, $\mathcal{A}_4$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_4$ succeeds if $b = b^{'}$. and hence $\mathcal{F}$ also succeeds Game IND-CCA.

Here we assume that $\mathcal{A}_4$ succeeds the game in polynomial time. But $\mathcal{F}$ simulates $\mathcal{A}_4$ and therefore $\mathcal{F}$ breaks the IND-CCA in polynomial time. Hence ${Adv_{\mathcal{A}_4}}^{TD-IND}(k) \leq {Adv_{\mathcal{F}.TE}}^{IND-CCA}(k)$ and that ${Adv_{\mathcal{A}_4}}^{TD-IND(k)}$ is equivalent to ${Adv_{\mathcal{F}.TE}}^{IND-CCA}(k)$.
We conclude that the proof of the lemma is completed.

LEMMA 5. Here we assume that $\mathcal{A}_5$ breaks TD-IND with advantage ${Adv_{\mathcal{A}_5}}^{TD-IND}(k)$, then their exists another adversary $\mathcal{B}$ which breaks the ANON-sID-CPA of IBE.

Proof. Here the adversary $\mathcal{B}$ will simulate the adversary $\mathcal{A}_5$ for breaking the ANON-sID-CPA of IBE.

1) Setup: Here $\mathcal{A}_5$ generates its public and private keys as $pk_S, sk_S$ respectively and sends $pk_S$ to $\mathcal{C}$. $\mathcal{C}$ also generates its public and private keys as $pk_R, sk_R$ respectively and sends $pk_R$ to $\mathcal{A}_5$.
2) Phase1: Here $\mathcal{A}_5$ is allowed to send a trapdoor query.Whenever $\mathcal{A}_5$ sends a trapdoor query for keyword $W \in \mathcal{W}$ to $\mathcal{B}$, $\mathcal{B}$ after receiving W sends ID = W to IBE.Extract and recovers $sk_U$ of ID. Now $\mathcal{B}$ generates trapdoor $T$ = TE.Encrypt$(SP, pk_S, sk_U)$ and sends it to $\mathcal{A}_5$.
3) Challenge: $\mathcal{A}_5$ sends two keywords ${W_0}^*, {W_1}^*$ for challenge to $\mathcal{B}$. But condition here is that ${W_0}^*, {W_1}^*$ should not have been sent as Trapdoor query. Now $\mathcal{B}$ sends ${ID_0}^* = {W_0}^*, {ID_1}^* = {W_1}^*$ for challenge to $\mathcal{C}$.In return, $\mathcal{C}$ forms a trapdoor $T^*$ for the keyword ${W_b}^*$ where $b \in \{0, 1\}$ is randomly chosen and sends to $\mathcal{B}$. Again, $\mathcal{B}$ sends this $T^*$ to $\mathcal{A}_5$.
4) Phase 2:$\mathcal{A}_5$ is again allowed to send trapdoor query with a condition that the keywords ${W_0}^*, {W_1}^*$ should not be queried.
5) Guess: Finally, $\mathcal{A}_5$ outputs its result as $b^{'} \in \{0, 1\}$. $\mathcal{A}_5$ succeeds if $b = b^{'}$. and hence $\mathcal{B}$ also succeeds Game $ANON - SID - CPA$.

Here we assume that $\mathcal{A}_5$ succeeds the game in polynomial time. But $\mathcal{B}$ simulates $\mathcal{A}_5$ and therefore $\mathcal{B}$ breaks the ANON-sID-CPA in polynomial time. Hence ${Adv_{\mathcal{A}_5}}^{SC-IND}(k) \leq {Adv_{B.IBE}}^{ANON-sID-CPA}(k)$ and that ${Adv_{\mathcal{A}_5}}^{SC-IND}(k)$ is equivalent to ${Adv_{B.IBE}}^{ANON-sID-CPA}(k)$.
We conclude that the proof of the lemma is completed.

### C. Comparison

The scheme described in [3] provides security for SC-IND if we consider adversary as the server only. The scheme neither provides OC-IND, nor secure for TD-IND. Later enhanced security model in [4] SC-IND and claimed to provide security for SC-IND with server as well as receiver as the adversary. Scheme in [7] which provides security for SC-IND for both server and receiver as adversary, also secure for TD-IND if we consider adversary to be any outsider other then server and receiver, but the scheme is insecure for OC-IND. Chen [9] proposed a scheme which is secure for SC-IND for both server and receiver as adversary, also secure for TD-IND considering outsider as adversary, and also provides security for OC-IND. Compared to above schemes our scheme enhances the security model TD-IND, and claimes to provide security if we assume adversary as outsider as well as receiver, also the scheme is secure for both SC-IND(sender and receiver as adversary) and OC-IND.

## V. Conclusion

To improve the security of dPEKS scheme(which was prone to online keyword guessing attack), recently a new scheme secure server-designation public key encryption with keyword search (SPEKS) was proposed. But the scheme still suffers from the offline keyword guessing attack launched by the malicious server, also the security model for TD-IND in SPEKS is incomplete.Our scheme overcomes ON-KGA, also the existing security models are enhanced for trapdoor indistinguishability by defining a new security model. Finally, we introduce a new scheme which is suitable for both cloud and email environment.

## References


[1] E. Kushilevitz and R. Ostrovsky, Replication is not needed single database, computationally-private information retrieval, in Proceedings of the 38th Annual Symposium on Foundations of Computer Science, ser. FOCS 97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 364. [Online]. Available: http://dl.acm.org/citation.cfm?id=795663.796363

[2] D. X. Song, D. Wagner, and A. Perrig, Practical techniques for searches on encrypted data, in Proceedings of the 2000 IEEE Symposium on Security and Privacy, ser. SP 00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 44. [Online]. Available: http://dl.acm.org/citation.cfm?id=882494.884426

[3] D. Boneh, G. D. 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, Proceedings, ser. Lecture Notes in Computer Science, vol. 3027. Springer, 2004, pp. 506522. [Online]. Available: http://www.iacr.org/cryptodb/archive/2004/EUROCRYPT/1954/1954.pdf

[4] J. Baek, R. Safavi-Naini, and W. Susilo, Public Key Encryption with Keyword Search Revisited. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 12491259.

[5] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, Off-Line Keyword Guessing Attacks on Recent Keyword Search Schemes over Encrypted Data. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 7583.

[6] E. Shen, E. Shi, and B. Waters, Predicate Privacy in Encryption Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 457473.

[7] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, Trapdoor security in a searchable public-key encryption scheme with a designated tester, J. Syst. Softw., vol. 83, no. 5, pp. 763771, May 2010. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2009.11.726

[8] W. Yau, R. C. Phan, S. Heng, and B. Goi, Keyword guessing attacks on secure searchable public key encryption schemes with a designated tester, Int. J. Comput. Math., vol. 90, no. 12, pp. 25812587, 2013.

[9] Y. Chen, SPEKS: secure server-designation public key encryption with keyword search against keyword guessing attacks, Comput. J., vol. 58, no. 4, pp. 922933, 2015.

[10] X.-J. Lin, L. Sun, and H. Qu, New framework for secure serverdesignation public key encryption with keyword search, Cryptology ePrint Archive, Report 2016/346, 2016, http://eprint.iacr.org/2016/346.