J Internet Serv Appl (2012) 3:233–241
DOI 10.1007/s13174-012-0063-1

ORIGINAL PAPER

Security models for delegated keyword searching within encrypted contents

Wei-Chuen Yau · Raphael C.-W. Phan · Swee-Huay Heng · Bok-Min Goi

Received: 23 April 2011 / Accepted: 11 June 2012 / Published online: 28 July 2012

Abstract Within modern internet infrastructure including networks that are ubiquitous, there is often a need for delegatable communication between nodes without compromising the confidentiality of information. In practice, this should be enforced while allowing some basic functionality for intermediate delegated nodes such as searching through encrypted content. This can be achieved using a Public key encryption with keyword search (PEKS) scheme, first proposed by Boneh et al., which enables to search publicly encrypted messages for keywords without revealing any information about the message. The issue of PEKS schemes being vulnerable to keyword guessing attacks (KGAs) was first shown by Byun et al., and two of the most recent PEKS schemes, i.e., due to Rhee et al. in (ASIACCS, pp 376–379, 2009; IEICE Electron Express 6(5):237–243, 2009) and (J Syst Softw 83(5):763–771, 2010), respectively, are designed with this security in mind. In this paper, we treat this KGA problem in detail and define new security models to capture KGAs against PEKS and designated PEKS schemes. These models are more security sufficient than the model considered by Rhee et al. (J Syst Softw 83(5):763–771, 2010); indeed the latter model does not afford sufficient adversarial capability in the sense that it is much weaker than the adversarial capability considered in the original IND-CKA model of Boneh et al. Our new models allow to capture KGAs on three recent designated PEKS schemes that cannot be captured in the weaker model of Rhee et al.

Keywords Modern internet · Security · Delegation · Public-key encryption with keyword search · Adversarial model · Guessing attack

R.C.-W. Phan carried out this part of the work while visiting Multimedia University.

Research funded by the Ministry of Science, Technology & Innovation under grant no. MOSTI/BGM/R&D/500-2/8, and by the Penang ICT Grant under grant no. MMU/RMC/PenangICT-2011/003.

W.-C. Yau
Faculty of Engineering, Multimedia University,
Cyberjaya, Malaysia
e-mail: wcyau@mmu.edu.my

R. C.-W. Phan (✉)
School of Electronic, Electrical and Systems Engineering,
Loughborough University, Leicestershire LE11 3TU, UK
e-mail: raphaelphan.crypt@gmail.com; r.phan@lboro.ac.uk

S.-H. Heng
Faculty of Information Science and Technology,
Multimedia University, Melaka, Malaysia
e-mail: shheng@mmu.edu.my

B.-M. Goi
Faculty of Engineering and Science,
Universiti Tunku Abdul Rahman, Kuala Lampur, Malaysia
e-mail: goibm@utar.edu.my

1 Introduction

Within the distributed and ubiquitous nature of modern internet, where delegation of tasks often occurs among nodes, it is crucial that this delegation is performed while retaining some form of security. One issue requiring attention is the need to ensure confidentiality of content as it traverses across nodes within such networks, while still allowing due to efficiency reasons for intermediary nodes to be delegated with the task to search for certain keywords.

Public key encryption with keyword search (PEKS) schemes, first proposed in [4], can be useful in this respect. Such a scheme allows a user to delegate searching capabilities on publicly encrypted data to a third party (e.g., an email storage provider) without revealing information in the data.

Consider a distributed e-mail system (or a distributed storage system) that consists of three entities, namely an originating sender, a receiver (R), and an email server or storage provider (S). The sender encrypts an email message with a conventional public key encryption scheme (PKE) [9]. S/He also encrypts a keyword (or a list of keywords) associated with the mail content with a PEKS scheme. S/He then appends the PEKS ciphertext to the PKE ciphertext and sends both to the email storage provider. R sends a trapdoor associated with a searching keyword w to the email storage provider S for retrieving all the emails containing the keyword. The PEKS scheme enables S to test whether w is a keyword associated with the email but learns nothing else about the email.

Boneh et al.'s PEKS scheme assumes a secure channel for sending a trapdoor, as observed by Baek et al. [3]. Otherwise, an eavesdropper who captures a trapdoor can run the test function (which is public) to determine the relation between a PEKS ciphertext and the trapdoor. Baek et al. [3] thus proposed a secure channel free PEKs (SCF-PEKS) scheme to remove the requirement of such a secure channel. This scheme is also known as a PEKS scheme with a designated server (dPEKS). A dPEKS scheme ensures that no one except the designated server is able to run the test function (`dTest`) since `dTest` is by design a function of the designated server's private key. Recently, some further dPEKS schemes have been proposed i.e., [6,10–12] that are either improving the security or efficiency of the Baek et al. [3] scheme.

Byun et al. [5] first addressed the vulnerability of keyword guessing attacks (KGA) on PEKS schemes. It is known that the keyword space of a keyword used by any PEKS scheme is usually small [4] in view of the highly redundant nature of our human language used to represent keywords, e.g., English word. In addition, users tend to query for a small set of keywords, for example, a user may search for some commonly used keywords such as "Urgent" in the "Subject" field of an email. In this case, adversaries are able to exploit this to exhaustively guess some candidate keywords and verify the correctness of their guesses. This results in the leak of information pertaining to encrypted emails. Byun et al. showed that the PEKS schemes of [4] and [8] are susceptible to the attacks. Yau et al. [14] later showed some similar attacks on dPEKS and PEKS schemes of [2,3]. In a different vein, Jeong et al. [7] showed that consistency implies insecurity to keyword guessing attacks in PEKS, where consistency means that it is not possible to find two keywords such that the `Test` function states that the PEKS ciphertext of one keyword and the trapdoor for the other keyword contain the same keyword.

To the best of our knowledge, the only security model defined in literature for PEKS against KGA is by Rhee et al. [11]. The other PEKS model is with respect to the indistinguishability goal under chosen keyword attacks (`IND-CKA`) (e.g., see [4]). This latter model by definition was not designed to capture KGA and primarily differs in the adversarial goal. Whereas, KGA models have the adversary aim to mount keyword guessing (KG) as a goal, IND has the adversary aiming to break an indistinguishability goal, i.e., being able to distinguish which one of two keywords was used as input to generate a PEKS ciphertext.

In this paper, we define models for KGA for the different settings of PEKS and dPEKS; these models achieve stronger KG notions than the KGA model of [11], and involve adversarial capability that is more analogous to the adversarial capability of the original `IND-CKA` model defined by Boneh et al. In contrast, the adversarial capability considered in the Rhee et al. model is much weaker than that of the adversary in the Boneh et al. `IND-CKA` model. Thus, a security proof within the Rhee et al. KGA model may not be able to fully capture security against KGA in view of its restrictive adversary definition. We then show that our models can capture attacks that exist on three recent dPEKS schemes of [10–12] that would otherwise not have been captured by the weaker model of Rhee et al.

2 Preliminaries

2.1 PEKS and dPEKS definitions

Definition 1 A PEKS scheme is defined by the following algorithms:

- `KeyGen`(s): On input a security parameter s, it returns a public–private key pair $\langle pk, sk\rangle$.
- `PEKS`(pk, w): On input a public key pk and a keyword w, it returns a PEKS ciphertext C.
- `Trapdoor`(sk, w): On input a private key sk and a keyword w, it returns a trapdoor T_w.
- `Test`(pk, C, T_w): On input a public key pk, a PEKS ciphertext C=`PEKS`(pk, w') and a trapdoor T_w= `Trapdoor`(sk, w), it returns 1 if $w = w'$ and 0 otherwise.

The security guarantees that an active adversary who is able to obtain trapdoors T_w for any w of his choice should not be able to distinguish between two encryptions for keywords of his choice for which he did not obtain the trapdoor. Formally, the indistinguishability under chosen keyword attack (`IND-CKA`) notion is defined by the following game between a challenger and the adversary $\mathcal{A}$.

1. The challenger runs the `KeyGen`(s) algorithm to generate the private and public key pair sk and pk. It keeps sk and gives pk to the adversary $\mathcal{A}$.

2. $\mathcal{A}$ queries the trapdoor generation oracle Trapdoor to obtain the trapdoor $T_w = \texttt{Trapdoor}(sk, w)$ for the corresponding keyword w of his choice.
3. $\mathcal{A}$ chooses two keywords w_0, w_1 and sends these to the challenger. The challenger flips a coin b and returns C^*=PEKS(pk, w_b).
4. $\mathcal{A}$ can adaptively query the Trapdoor oracle to obtain the trapdoor T_w=Trapdoor(sk, w) for any input keyword w of his choice, where $w \neq \{w_0, w_1\}$.
5. $\mathcal{A}$ outputs a bit b' as his guess of the bit b.

$\mathcal{A}$ wins this IND-CKA game if $b = b'$. Thus, we define $\mathcal{A}$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{PEKS-IND-CKA}}(s) = \left|\Pr[b = b'] - \frac{1}{2}\right|$$

Definition 2 A PEKS scheme is said to be IND-CKA secure if $\mathsf{Adv}_{\mathcal{A}}^{\text{PEKS-IND-CKA}}(s)$ is negligible in the security parameter.

The consistency of a PEKS scheme is defined by the following game between an attacker $\mathcal{A}$ and a challenger [1]:

1. The challenger runs the KeyGen(s) algorithm to generate the private and public key pair sk and pk. It keeps sk and gives pk to $\mathcal{A}$.
2. $\mathcal{A}$ sends the challenger two keywords $w, w' \in \{0, 1\}^*$.
3. PEKS ciphertext C = PEKS(pk, w) and trapdoor $T_{w'}$ = Trapdoor(sk, w') are computed.
4. If $w \neq w'$ and Test($pk, C, T_{w'}$) = 1, then return 1, else return 0.

The attacker wins this game if it can make the consistency fail. We define $\mathcal{A}$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{PEKS-consist}}(s) = \Pr\left[\mathsf{Test}(pk, \mathsf{PEKS}(pk, w), \mathsf{Trapdoor}(sk, w')) = 1\right]$$

Definition 3 We say that a PEKS scheme is perfectly consistent if this advantage is 0 for all (computationally unrestricted) adversaries $\mathcal{A}$; statistically consistent if it is negligible for all (computationally unrestricted) adversaries $\mathcal{A}$; and computationally consistent if it is negligible for all polynomial time bounded adversaries $\mathcal{A}$.

Definition 4 A dPEKS scheme (or Secure Channel Free PEKS) is defined by the following algorithms:

- GlobalSetup(s): On input a security parameter s, it returns global parameter, $\mathcal{GP}$.
- KeyGen$_S$($\mathcal{GP}$): On input $\mathcal{GP}$, it returns a public–private key pair $\langle pk_S, sk_S\rangle$ of server S.
- KeyGen$_R$($\mathcal{GP}$): On input $\mathcal{GP}$, it returns a public–private key pair $\langle pk_S, sk_S\rangle$ of receiver R.
- dPEKS($\mathcal{GP}$, pk_R, pk_S, w): On input $\mathcal{GP}$, pk_R, pk_S, and a keyword w, it returns a dPEKS ciphertext C of w.
- Trapdoor($\mathcal{GP}, sk_R, w$): On input $\mathcal{GP}$, sk_R, and a keyword w, it returns a trapdoor T_w.
- dTest($\mathcal{GP}, C, sk_S, T_w$): On input $\mathcal{GP}$, sk_S, a dPEKS ciphertext C = dPEKS($\mathcal{GP}$, pk_R, pk_S, w'), and a trapdoor T_w = Trapdoor($\mathcal{GP}, sk_R, w$), it returns 1 if $w = w'$ and 0 otherwise.

The security of a dPEKS scheme guarantees that a malicious server who obtains trapdoors for any non-challenged keywords should not be able to distinguish between ciphertexts associated with the challenged keywords w_0, w_1 of his choice; while an external attacker (including the receiver) who does not has the private key of the server should not be able to distinguish between the challenged ciphertext of two challenged keyowrds of his choice, even though he can obtain test results for any ciphertexts (where none of the elements of the ciphertext are identical to the one in challenged ciphertext) and trapdoors associated with non-challenged keywords. We formally define the indistinguishability under chosen keyword attack (IND-CKA) notion of dPEKS with the following two games between a challenger and the adversary $\mathcal{A}_i$ ($i = 1, 2$).

Game 1 $\mathcal{A}_1$ is assumed to be a malicious server.

1. The challenger $\mathcal{B}$ runs the $\mathsf{KeyGen}_R(s)$ algorithm to generate the receiver's public–private key pair (pk_R, sk_R) and gives pk_R to $\mathcal{A}_1$.$\mathcal{A}_1$ generates the server's public–private key pair $\langle pk_S, sk_S\rangle$ and gives pk_S to $\mathcal{B}$.
2. $\mathcal{A}_1$ queries the trapdoor generation oracle Trapdoor to obtain the trapdoor $T_w = \mathsf{Trapdoor}(\mathcal{GP}, sk_R, w)$ for the corresponding keyword w of his choice.
3. $\mathcal{A}_1$ chooses two keywords w_0, w_1 and sends these to the challenger. The challenger picks a random $b \in \{0, 1\}$ and returns $C^* = \texttt{PEKS}(\mathcal{GP}, pk_R, pk_S, w_b)$.
4. $\mathcal{A}_1$ can adaptively query the Trapdoor oracle to obtain the resulting trapdoor T_w for any input keyword w of his choice, where $w \neq \{w_0, w_1\}$.
5. $\mathcal{A}_1$ outputs a bit $b' \in \{0, 1\}$ as his guess of the bit b.

$\mathcal{A}_1$ wins **Game 1** if $b = b'$. Thus, we define $\mathcal{A}_1$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}_1}^{\text{dPEKS-IND-CKA}}(s) = \left|\Pr[b = b'] - \frac{1}{2}\right|$$

Game 2 $\mathcal{A}_2$ is assumed to be an external attacker (including a malicious receiver).

1. The challenger $\mathcal{B}$ runs the $\mathsf{KeyGen}_S(s)$ algorithm to generate the server's public–private key pair (pk_S, sk_S)

and gives pk_S to $\mathcal{A}_2$.$\mathcal{A}_2$ generates the receiver's public–private key pair $\langle pk_R, sk_R\rangle$ and gives pk_R to $\mathcal{B}$.
2. $\mathcal{A}_2$ queries the `dTest` oracle to obtain the test results for any ciphertext C and trapdoor T_w of his choice.
3. $\mathcal{A}_2$ chooses two keywords w_0, w_1 and sends these to the challenger. The restriction is that $\mathcal{A}_2$ did not previously ask the `dTest` oracle for the test result of the trapdoors T_{w_0} or T_{w_1}. The challenger picks a random $b \in \{0, 1\}$ and returns $C^* = \texttt{dPEKS}(\mathcal{GP}, pk_R, pk_S, w_b)$.
4. $\mathcal{A}_2$ can adaptively query the `dTest` oracle to obtain the test result for any ciphertext C and trapdoor T_w for any input keyword w of his choice, as long as the corresponding elements of C is not the same as one of C^* and $w \neq \{w_0, w_1\}$.
5. $\mathcal{A}_2$ outputs a bit $b' \in \{0, 1\}$ as his guess of the bit b.

$\mathcal{A}_2$ wins **Game 2** if $b = b'$. Thus, we define $\mathcal{A}_2$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}_2}^{\text{dPEKS-IND-CKA}}(s) = \left|\Pr[b = b'] - \frac{1}{2}\right|$$

Definition 5 A dPEKS scheme is said to be `IND-CKA` secure if for any polynomial time adversary $\mathcal{A}_i$ $(i = 1, 2)$, we have that $\mathsf{Adv}_{\mathcal{A}_i}^{\text{dPEKS-IND-CKA}}(s)$ is negligible in the security parameter.

The consistency of a dPEKS scheme is defined by the following game between an attacker $\mathcal{A}$ and a challenger which is similar to Definition 3:

1. The challenger runs the $\mathsf{KeyGen}_S(s)$ and $\mathsf{KeyGen}_R(s)$ algorithms to generate the public–private key pairs $\langle pk_S, sk_S\rangle$ and $\langle pk_R, sk_R\rangle$ of a server and a receiver, respectively. It keeps sk_S and sk_R, gives pk_S and pk_R to $\mathcal{A}$.
2. $\mathcal{A}$ sends the challenger two keywords $w, w' \in \{0, 1\}^*$.
3. dPEKS ciphertext $C = \texttt{dPEKS}(\mathcal{GP}, pk_R, pk_S, w)$ and trapdoor $T_{w'} = \texttt{Trapdoor}(\mathcal{GP}, sk_R, w')$ are computed.
4. If $w \neq w'$ and $\texttt{dTest}(\mathcal{GP}, C, sk_S, T_{w'}) = 1$, then return 1, else return 0.

The attacker wins this game if it can make the consistency fail. We define $\mathcal{A}$'s advantage as

$$\begin{aligned}\mathsf{Adv}_{\mathcal{A}}^{\text{dPEKS-consist}}(s) = \Pr[&\mathsf{dTest}(\mathcal{GP}, sk_S,\\ &\mathsf{dPEKS}(\mathcal{GP}, pk_R, pk_S, w),\\ &\mathsf{Trapdoor}(\mathcal{GP}, sk_R, w')) = 1]\end{aligned}$$

Definition 6 We say that a dPEKS scheme is perfectly consistent if this advantage is 0 for all (computationally unrestricted) adversaries $\mathcal{A}$; statistically consistent if it is negligible for all (computationally unrestricted) adversaries $\mathcal{A}$; and computationally consistent if it is negligible for all polynomial time bounded adversaries $\mathcal{A}$.

2.2 Recent dPEKS schemes

To differentiate the schemes, we call the two recently proposed dPEKS schemes of [10,12] as RPSL and RSK, respectively. Note that a subsequent dPEKS scheme appeared in [11] recently, but it is a slight variant of RSK so without loss of generality, our discussions on RSK apply to this variant as well. Let $\mathbb{G}, \mathbb{G}_T$ be groups of prime order p and g be a generator of $\mathbb{G}$. We denote s as a security parameter and $H_1 : \{0, 1\}^* \to \mathbb{G}$, $H_2 : \mathbb{G}_T \to \{0, 1\}^s$ as two hash functions. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

1. Bilinear: For all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degenerate: $e(g, g) \neq 1$.
3. Computable: There is an efficient algorithm to compute $e(u, v) \in \mathbb{G}_T$.

2.2.1 RPSL scheme

Rhee et al. [10] proposed a dPEKS scheme, along with an enhanced `IND-CKA` security model that builds on that of [3]. RPSL did not make any security claims against KGA resistance, although its security is proven within a strong `IND-CKA` model with a powerful adversary who has access to `Trapdoor` and `dTest` oracles. This RPSL dPEKS scheme is described as follows:

- $\texttt{GlobalSetup}(s)$: Given a security parameter s, it returns a global parameter $\mathcal{GP} = \langle \mathbb{G}, \mathbb{G}_T, e, H_1(\cdot), H_2(\cdot), g, h, u, v\rangle$, where $h, u, v \in \mathbb{G}$ are random values.
- $\mathsf{KeyGen}_S(\mathcal{GP})$: Takes as input $\mathcal{GP}$, chooses a random $x \in \mathbb{Z}_P^*$ and sets $sk_S = x$. Computes $pk_S = \langle pk_{S,1}, pk_{S,2}, pk_{S,3}\rangle = \langle g^x, h^{\frac{1}{x}}, u^{\frac{1}{x}}\rangle$. Outputs $\langle pk_S, sk_S\rangle$ to the server S and publishes pk_S.
- KeyGen_R $(\mathcal{GP})$: Takes as input $\mathcal{GP}$, chooses a random $y \in \mathbb{Z}_P^*$ and sets $sk_R = y$. Computes $pk_R = \langle pk_{R,1}, pk_{R,2}, pk_{R,3}\rangle = (g^y, h^{\frac{1}{y}}, v^y\rangle$. Outputs $\langle pk_R, sk_R\rangle$ to the receiver R and publishes pk_R.
- $\texttt{dPEKS}(\mathcal{GP}, pk_R, pk_S, w)$: Takes as input $\mathcal{GP}$, receiver's public key $pk_R = \langle pk_{R,1}, pk_{R,2}, pk_{R,3}\rangle$, a server's public key $pks = \langle pk_{S,1}, pk_{S,2}, pk_{S,3}\rangle$, and a keyword w. This algorithm checks if $e(pk_{S,1}, pk_{S,2}) = e(g, h), e(pk_{S,1}, pk_{S,3}) = e(g, u), e(pk_{R,1}, pk_{R,2}) = e(g, h)$, and $e(pk_{R,2}, pk_{R,3}) = e(h, v)$. If any of these conditions is false, this algorithm stops. Otherwise, this algorithm chooses a random value $r \in \mathbb{Z}_p^*$ and computes a dPEKS ciphertext $C = \langle pk_{R,1}^r, H_2(e(pk_{S,1}, H_1(w)^r))\rangle$.
- $\texttt{Trapdoor}(\mathcal{GP}, sk_R, w)$: Takes as input $\mathcal{GP}$, a receiver's private key $sk_R = y$, and a keyword w. Computes and outputs $T_w = H_1(w)^{\frac{1}{y}}$.

- dTest($\mathcal{GP}, C, sk_S, T_w$): Takes as input $\mathcal{GP}, C = \langle A, B\rangle$, a private key of server $sk_S = x$, and a trapdoor T_w. This algorithm checks if $B = H_2(e(A, T_w^x))$. If the above equality is satisfied, then it outputs "1"; otherwise, it outputs "0".

2.2.2 RSK-like schemes

Rhee et al. [12] proposed a dPEKS scheme to be resistant to KGAs. The proof sketch was constructed without defining a security model; the gist of the security argument is that the adversary cannot perform keyword guessing attacks because s/he has no knowledge of the private key sk_S of the server nor the private key sk_R of the receiver. A subsequent variant appeared in [11] which is slightly different, this one had its security proven within an explicit KGA model. For the rest of this paper, our discussion on RSK also applies to its variant in [11]. The RSK dPEKS scheme is described as follows:

- Global Setup(s): Given a security parameter s, it returns a global parameter $\mathcal{GP} = \langle \mathbb{G}, \mathbb{G}_T, e, H_1(\cdot), H_2(\cdot), g, \mathcal{KS}\rangle$, where $\mathcal{KS}$ is a keyword space.
- KeyGen$_S(\mathcal{GP})$: Takes as input $\mathcal{GP}$, chooses a random $\alpha \in \mathbb{Z}_P^*$ and $Q \in \mathbb{G}$, and returns server's public–private key pair $sk_S = \alpha$ and $pk_S = \langle \mathcal{GP}, Q, y\rangle = (\mathcal{GP}, Q, g^\alpha\rangle$.
- KeyGen$_R$ ($\mathcal{GP}$): Takes as input $\mathcal{GP}$, chooses a random $x \in \mathbb{Z}_P^*$ and returns $sk_R = x$ and $pk_R = g^x$ as receiver's private and public keys, respectively.
- dPEKS(pk_R, pk_S, w): Chooses a random value $r \in \mathbb{Z}_P^*$ and outputs $C = \langle A, B\rangle = \langle (pk_R)^r, H_2(e(y, H_1(w)^r))\rangle$, where $w \in \mathcal{KS}$.
- Trapdoor(sk_R, w): Takes as input a receiver's secret key sk_R, a keyword w. Chooses a random value $r' \in \mathbb{Z}_P^*$. Computes and outputs $T_w = \langle T_1, T_2\rangle = \langle y^{r'}, H_1(w)^{\frac{1}{x}} \cdot g^{r'}\rangle$, where $w \in \mathcal{KS}$.
- dTest($\mathcal{GP}, C, sk_S, T_w$): Takes as input $C = \langle A, B\rangle$, a secret key of server sk_S, and a trapdoor T_w. This algorithm computes $\mathcal{T} = \frac{T_2^\alpha}{T_1}$ and checks if $B = H_2(e(A, \mathcal{T}))$. If the above equalities are satisfied, then output "1"; otherwise, output "0".

3 Adversarial models for (d)PEKS

A security model defines the adversary's capability and the goal that the adversary aims to circumvent; the model is then parametrized by this goal-capability pair, i.e. the IND-CCA model defines the indistinguishability goal against a chosen-ciphertext querying adversary.

PEKS was fundamentally designed to solve the problem of insider adversaries (the server gateway) with the ability to perform chosen-keyword queries (CKA) aiming to break the IND goal. The original IND-CKA model for PEKS allowed the adversary oracle access to the Trapdoor function (to model chosen-keyword queries) and the adversary was able to run the Test function since it is public. The challenge posted to the adversary corresponding to the IND goal consisted of a challenge PEKS ciphertext C^* produced by PEKS-encrypting one of two keywords.

The subsequent IND-CKA dPEKS models of [10,11] also considered insider adversaries with oracle access to the Trapdoor and dTest functions. The latter access in order to retain equivalent adversarial capability in the transition from PEKS to dPEKS since the dTest function by design is private in contrast to the public Test function in PEKS.

4 Security models for (d)PEKS against KG−CKA

In this section, we propose security models for (d)PEKS in the sense of keyword guessing resistance under chosen keyword attacks (KG-CKA). In particular, the KG-sCKA dPEKS model of Sect. 4.3 defines the adversary with capability equivalent to that of the IND-CKA PEKS model and the IND-CKA dPEKS model of [10,11] respectively, i.e., the adversary has oracle access to both the Trapdoor and dTest functions. This also corroborates well with the result of [7] that showed consistency implies impossibility against KGA, where the implication proof required that the KG adversary has access to the Test function and be given the challenge ciphertext C^* and challenge trapdoor T_w^*.

4.1 The KG-CKA PEKS model

To model the KG-CKA PEKS notion for conventional PEKS, we define the following game between an adversary $\mathcal{A}$ and a challenger.

1. The challenger runs the KeyGen(s) algorithm to generate a public and private key pair $\langle pk, sk\rangle$. It keeps sk and gives pk to $\mathcal{A}$.
2. $\mathcal{A}$ asks the challenger to send the challenge. The challenger randomly selects $w^* \in \{0,1\}^*$. The challenger generates a PEKS ciphertext $C^* = \mathsf{PEKS}(pk, w^*)$ and a trapdoor $T_w^* = \mathsf{Trapdoor}(sk, w^*)$ such that $\mathsf{Test}(C^*, T_w^*) = 1$. It returns the challenge $\langle C^*, T_w^*\rangle$ to $\mathcal{A}$.
3. $\mathcal{A}$ can query the trapdoor generation oracle Trapdoor $(sk, \cdot)$ to obtain the trapdoors corresponding to keywords $w \in \{0,1\}^*$ of his choice.
4. $\mathcal{A}$ outputs his guess w'.

The adversary wins this game if s/he can guess the keyword correctly, i.e., $w^* = w'$. We define $\mathcal{A}$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{KG-CKAPEKS}}(s) = \Pr[w^* = w'].$$

Definition 7 A PEKS scheme is said to be KG-CKA-secure if $\mathsf{Adv}_{\mathcal{A}}^{\text{KG-CKAPEKS}}(s)$ is negligible in the security parameter s.

Note that the adversary in this model has essentially the same capability as that of the [4] IND-CKA adversary.

4.2 The KG-wCKA dPEKS model

We now define the security model for dPEKS against KGAs. Recall that the main difference between a dPEKS and PEKS is that the dTest function in dPEKS is not public compared to the Test function in PEKS, i.e., dTest function requires as input the private key of the server. As such, the adversary will have stronger adversarial capabilities if the dTest function is accessible to the adversary as an oracle; and, therefore, based on this characterization, we define two security games denoted as weak and strong notions, respectively.

The weaker notion, KG-wCKA dPEKS, is defined by the following game between the adversary $\mathcal{A}$ and a challenger.

1. The challenger runs the GlobalSetup(s) algorithm to generate a global parameter $\mathcal{GP}$. The $\mathsf{KeyGen}_S(\mathcal{GP})$ algorithm and KeyGen_R $(\mathcal{GP})$ algorithm are run to generate public-private key pairs $\langle pk_S, sk_S\rangle$ and $\langle pk_R, sk_R\rangle$ of the server and receiver respectively. The generated $\mathcal{GP}$, pk_R, and pk_S are given to $\mathcal{A}$ while sk_R and sk_S are kept secret from $\mathcal{A}$.
2. $\mathcal{A}$ asks the challenger to send the challenge. The challenger randomly selects $w^* \in \{0,1\}^*$. The challenger generates a dPEKS ciphertext $C^* = \mathsf{dPEKS}(pk_R, pk_S, w^*)$ and a trapdoor $T_w^* = \mathsf{Trapdoor}(sk_R, w^*)$ such that $\mathsf{dTest}(\mathcal{GP}, C^*, sk_S, T_w^*) = 1$. It returns the challenge $\langle C^*, T_w^*\rangle$ to $\mathcal{A}$.
3. $\mathcal{A}$ can query the trapdoor generation oracle Trapdoor $(sk_R, \cdot)$ to obtain the trapdoor corresponding to any keyword $w \in \{0,1\}^*$ of his choice.
4. $\mathcal{A}$ outputs his guess w'.

The adversary wins this game if s/he can guess the keyword correctly, i.e., $w^* = w'$. We define $\mathcal{A}$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{KG-wCKAdPEKS}}(s) = \Pr[w^* = w']$$

Definition 8 A dPEKS scheme is said to be KG-wCKA-secure if $\mathsf{Adv}_{\mathcal{A}}^{\text{KG-wCKAdPEKS}}(s)$ is negligible in the security parameter s.

4.3 The KG-sCKA dPEKS model

The stronger notion, KG-sCKA dPEKS, is defined by the following game between the adversary $\mathcal{A}$ and a challenger.

1. The challenger runs the GlobalSetup(s) algorithm to generate a global parameter $\mathcal{GP}$. The KeyGen_S $(\mathcal{GP})$ algorithm and KeyGen_R $(\mathcal{GP})$ algorithm are run to generate public–private key pair $\langle pk_S, sk_S\rangle$ and $\langle pk_R, sk_R\rangle$ of the server and receiver, respectively. The generated $\mathcal{GP}$, pk_R, and pk_S are given to A while sk_R and sk_S are kept secret from $\mathcal{A}$.
2. $\mathcal{A}$ asks the challenger to send the challenge. The challenger randomly selects $w^* \in \{0,1\}^*$. The challenger generates a dPEKS ciphertext $C^* = \mathsf{dPEKS}(pk_R, pk_S, w^*)$ and a trapdoor $T_w^* = \mathsf{Trapdoor}(sk_R, w^*)$ such that $\mathsf{dTest}(\mathcal{GP}, C^*, sk_S, T_w^*) = 1$. It returns the challenge $\langle C^*, T_w^*\rangle$ to $\mathcal{A}$.
3. $\mathcal{A}$ can adaptively query the trapdoor oracle $\mathsf{Trapdoor}$ $(sk_S, \cdot)$ to obtain the trapdoor corresponding to any keyword $w \in \{0,1\}^*$ of his choice. $\mathcal{A}$ can also adaptively query the dTest oracle to obtain the result of $\mathsf{dTest}(\mathcal{GP}, C, sk_S, T_w)$ for the $\langle C, T_w\rangle$ of his choice.
4. $\mathcal{A}$ outputs his guess w'.

The adversary wins this game if s/he can guess the keyword correctly, i.e., $w^* = w'$. We define $\mathcal{A}$'s advantage as

$$\mathsf{Adv}_{\mathcal{A}}^{\text{KG-sCKAdPEKS}}(s) = \Pr[w^* = w'].$$

Definition 9 A dPEKS scheme is said to be KG-sCKA-secure if $\mathtt{Adv}_{\mathcal{A}}^{\text{KG-sCKAdPEKS}}(s)$ is negligible in the security parameter s.

Observe here that the adversary basically has the same capability as the adversary in the IND-CKA dPEKS games of [10,11]; in essence the two notions only differ in the adversarial goal, i.e. KG versus IND.

The first known dPEKS notions that consider characterizing the adversarial ability based on the fact that the dTest algorithm of dPEKS is not public unlike the Test algorithm of PEKS, are the IND-CKA dPEKS notions of [10,11]. This is similar to IND-CKA PEKS but where while PEKS has only one private algorithm, i.e., Trapdoor, dPEKS has two private algorithms, i.e., Trapdoor and dTest. And, therefore, the same treatment as applied to Trapdoor is rightly applicable to dTest as well, i.e. considering private algorithms as oracles accessible by the adversary.

4.4 On the KG-CKA dPEKS model of Rhee et al.

The KG-CKA dPEKS model of [11] defines a very restricted KG adversary, who is not given oracle access to the Trapdoor and dTest functions; it is, therefore, weaker

than the adversarial capability considered in the original IND-CKA model of Boneh et al. and weaker than even our KG-wCKA dPEKS model. Furthermore, it only returns the challenge trapdoor T_w^* to the adversary and not including the challenge ciphertext C^*. Thus, the Rhee et al. KG-CKA dPEKS model cannot capture the basic KG attack presented by [12] against the dPEKS scheme of [3] which requires the adversarial capability of having access to both the challenge trapdoor and the challenge ciphertext.

5 Capturing KGAs within the models

In this section we describe how the KG-CKA dPEKS models we discussed in the previous section can capture some KG attacks.

In comparison with the original KG attacks on PEKS schemes whose KG adversary has capability corresponding to the KG-CKA PEKS model where the KG adversary can perform the public Test function and has oracle access to the Trapdoor function; recall that in the Rhee et al.'s KG-CKA model, no adversarial oracle access is provided, while in the KG-wCKA dPEKS model the adversary is not given the dTest oracle. So both these dPEKS models are much more restricted in comparison to the KG-CKA PEKS model.

5.1 The RPSL scheme in the KG-CKA dPEKS model of Rhee et al.

We present a KG attack on the RPSL scheme in the KG-CKA dPEKS model of [11], where no oracle queries are given to the adversary.

Upon receiving the challenge $T_w^* = H_1(w^*)^{\frac{1}{y}}$, the adversary performs the attack in the following steps:

1. $\mathcal{A}$ selects a keyword $w_i \in \mathcal{KS}$, and computes $e(g, H_1(w_i))$.
2. If $e(pk_{R,1}, T_w^*) = e(g, H_1(w_i))$, $\mathcal{A}$ outputs $w' = w_i$.
3. Otherwise, goto step 1.

It is easy to show the correctness of equality in step 2 when $w^* = w_i$, i.e., $e(pk_{R,1}, T_w^*) = e(g^y, H_1(w^*)^{\frac{1}{y}}) = e(g, H_1(w^*)) = e(g, H_1(w_i))$.

Surprisingly, this attack does not work on the other recent dPEKS schemes. The problem that we exploit lies in the way the trapdoor is structured. More precisely, the private key influence in the trapdoor value can be cancelled out by the public key.

5.2 The RPSL scheme in the KG-wCKA dPEKS model

Even if the RPSL scheme did not have the flaw in its trapdoor structure that allowed the private key influence to be cancelled out, a KG attack still works by exploiting its deterministic Trapdoor function. This attack can be captured in our KG-wCKA model although it cannot be captured by the KG-CKA dPEKS model of Rhee et al. which restricts the adversary substantially, as discussed in Sect. 4.4.

1. $\mathcal{A}$ selects a keyword $w_i \in \mathcal{KS}$, and sends w_i to the $\mathtt{Trapdoor}(sk_S, \cdot)$ oracle to obtain the corresponding $T_{w_i} = H_1(w_i)^{\frac{1}{y}}$.
2. If $T_w^* = T_{w_i}$, *i.e.*, $H_1(w^*)^{\frac{1}{y}} = H_1(w_i)^{\frac{1}{y}}$, $\mathcal{A}$ outputs $w' = w_i$.
3. Otherwise, goto step 1.

The gist is that the output of the $\mathtt{Trapdoor}(sk_S, \cdot)$ oracle is deterministic. Thus, when $T_w^* = T_{w_i}$ in step 2, we have $\Pr[w^* = w_i] = \Pr[w^* = w'] = 1$ such that $\mathsf{Adv}_{\mathcal{A}}^{\text{KG-wCKAdPEKS}}(s) = \Pr[w^* = w'] = 1$. Since $\mathsf{Adv}_{\mathcal{A}}^{\text{KG-wCKAdPEKS}}(s)$ is non-negligible, the RPSL scheme is not KG-wCKA-secure.

In contrast, unlike the RPSL scheme cryptanalyzed above, we are unable to mount any attacks on the RSK scheme in the KG-wCKA model. The essence is that since the output of the RSK scheme's $\mathtt{Trapdoor}(sk_S, \cdot)$ oracle is probabilistic, the guess verification in step (2.) of the above attacks cannot be performed.

5.3 The Three dPEKS schemes in the KG-sCKA model

The nice thing about the [12,11] schemes are that they are designed with the aim to resist keyword guessing (KG) attacks, and come with proofs of this kind of KG security.

That said, the RSK paper, i.e., [12], did make reference to their IND-CKA dPEKS model, i.e., [10] when they discuss the IND-CKA dPEKS security. Their IND-CKA dPEKS notions considered both the Trapdoor and dTest as oracles to the adversary, so it makes sense to consider the KG security of the RSK scheme with similar adversarial ability.

In more detail, the RSK scheme (as is the RPSL scheme) being a dPEKS scheme rather than a conventional PEKS, by design, therefore, has its dTest function be private to the designated tester in contrast to a conventional PEKS scheme where the Test function is public and thereby doable by anyone. Hence, a dPEKS model that is more analogous to the KG-CKA PEKS model of its PEKS counterpart and that better preserves the adversarial setting in moving from PEKS to dPEKS, would appear to be the stronger KG-sCKA dPEKS model where the dTest function is also readily available to the adversary as an oracle query, as like in the KG-CKA

PEKS counterpart model where Test is public. Furthermore, this KG-sCKA dPEKS model corresponds well to the RSK designers' consideration of RSK's IND-CKA dPEKS security, where they use their IND-CKA dPEKS model including the dTest oracle available to the adversary.

Within this context, we show that the RSK scheme falls to KGA under the KG-sCKA dPEKS model. The RPSL scheme equally falls; for simplicity we only describe the attack on the RSK scheme.

More precisely, the challenger runs the GlobalSetup(s), KeyGen_S ($\mathcal{GP}$), and KeyGen_R ($\mathcal{GP}$) algorithms. The generated global parameter $\mathcal{GP} = \langle \mathbb{G}, \mathbb{G}_T, e, H_1(\cdot), H_2(\cdot), g, \mathcal{KS} \rangle$, server's public key $pk_S = \langle \mathcal{GP}, Q, y \rangle = (\mathcal{GP}, Q, g^\alpha \rangle$, and receiver's public key $pk_R = g^x$, are given to the adversary $\mathcal{A}$. Upon receiving the challenge $\langle C^*, T_w^* \rangle = \langle \langle (g^x)^r, H_2(e(g^\alpha, H_1(w^*)^r)) \rangle, \langle (g^\alpha)^{r'}, H_1(w^*)^{\frac{1}{x}} \cdot g^{r'} \rangle \rangle$, the adversary performs the attack in the following steps:

1. $\mathcal{A}$ selects a keyword $w_i \in \mathcal{KS}$, and computes $C_i = \langle \langle (g^x)^r, H_2(e(g^\alpha, H_1(w_i)^r)) \rangle$. Note that $\mathcal{A}$ can compute a fixed C_i by setting $r = 1$.
2. $\mathcal{A}$ sends $\langle C_i, T_w^* \rangle$ to the dTest oracle and obtains an output b from the oracle, where $b \in \{0, 1\}$.
3. If $b = 1$, $\mathcal{A}$ outputs $w' = w_i$.
4. Otherwise, goto step 1.

Since the RSK scheme is computationally consistent, see [12], the probability for both events $w_i \neq w^*$ and dTest($\mathcal{GP}$, C_i, sk_S, T_w^*) = 1 to occur is negligible, denoted by ϵ. If $w^* = w_i$, we have $\Pr[w^* = w_i] = \Pr[w^* = w'] = 1 - \epsilon$ such that $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{KG-sCKAdPEKS}}(s) = \Pr[w^* = w'] = 1 - \epsilon$. Since $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{KG-sCKAdPEKS}}(s)$ is non-negligible, the RSK scheme is not KG-sCKA-secure.

We remark that an alternative attack exists, as follows:

1. $\mathcal{A}$ selects a keyword $w_i \in \mathcal{KS}$, and sends w_i to the Trapdoor($sk_S, \cdot$) oracle to obtain the corresponding $T_{w_i} = \langle T_1, T_2 \rangle = \langle y^{r'}, H_1(w_i)^{\frac{1}{x}} \cdot g^{r'} \rangle$.
2. $\mathcal{A}$ sends $\langle C^*, T_{w_i} \rangle$ to the dTest oracle and obtains an output b from the oracle, where $b \in \{0, 1\}$.
3. If $b = 1$, $\mathcal{A}$ outputs $w' = w_i$.
4. Otherwise, goto step 1.

6 A KG−sCKA dPEKS scheme

To demonstrate the feasibility of the KG-sCKA dPEKS model, we apply the method in [13] and construct a KG-sCKA dPEKS scheme based on [12]. In this scheme, the same assumption is made as per [13] that there is an initialization phase so that a sender is required to register a keyword w with a receiver before the sender can generate a keyword ciphertext. During this keyword registration, the receiver computes $H_1(w||v)$ with her private key v, and the sender obtains $H_1(w||v)$ from the receiver to be used later to create the dPEKS ciphertext associated with keyword w. The scheme is described as follows:

- Global Setup(s): Given a security parameter s, it returns a global parameter $\mathcal{GP} = \langle \mathbb{G}, \mathbb{G}_T, e, H_1(\cdot), H_2(\cdot), g, \mathcal{KS} \rangle$, where $\mathcal{KS}$ is a keyword space.
- $\mathsf{KeyGen}_S(\mathcal{GP})$: Takes as input $\mathcal{GP}$, chooses a random $\alpha \in \mathbb{Z}_P^*$, and returns server's public–private key pair $sk_S = \alpha$ and $pk_S = \langle \mathcal{GP}, y \rangle = (\mathcal{GP}, g^\alpha \rangle$.
- KeyGen_R ($\mathcal{GP}$): Takes as input $\mathcal{GP}$, chooses random $u, v \in \mathbb{Z}_P^*$ and returns $sk_R = \langle u, v \rangle$ and $pk_R = g^u$ as receiver's private and public keys, respectively.
- dPEKS(pk_R, pk_S, w'): Takes as input the public keys of server and receiver, and $w' = \langle w, H_1(w||v) \rangle$ where $w \in \mathcal{KS}$ and $H_1(w||v)$ was obtained by the server from the receiver during an initial keyword registration phase. Chooses a random value $r \in \mathbb{Z}_P^*$ and outputs $C = \langle A, B \rangle = \langle (pk_R)^r, H_2(e(y, H_1(w||v)^r)) \rangle$.
- Trapdoor(sk_R, w): Takes as input a receiver's secret key sk_R, a keyword w. Chooses a random value $r' \in \mathbb{Z}_P^*$. Computes and outputs $T_w = \langle T_1, T_2 \rangle = \langle y^{r'}, H_1(w||v)^{\frac{1}{u}} \cdot g^{r'} \rangle$, where $w \in \mathcal{KS}$.
- dTest($\mathcal{GP}$, C, sk_S, T_w): Takes as input $C = \langle A, B \rangle$, a secret key of server sk_S, and a trapdoor T_w. This algorithm computes $\mathcal{T} = \frac{T_2^\alpha}{T_1}$ and checks if $B = H_2(e(A, \mathcal{T}))$. If the above equalities are satisfied, then output "1"; otherwise, output "0".

Table 1 Comparison of schemes with respect to different models

Schemes	Security models		
	[11]	KG-wCKA	KG-sCKA
RPSL [10]	No	No	No
RSK-like [11,12]	Yes	Yes	No
KG-sCKA dPEKS of Sect. 6	Yes	Yes	Yes

The dPEKS scheme is KG-sCKA secure at the expense of additional interaction between the sender and the receiver. To see this, the crucial point is to analyze the additional dTest oracle available to the adversary in the KG-sCKA model. With this oracle access, the adversary can issue black box queries to the dTest function under the unknown private key sk_S. Nevertheless, even with this, the oracle can only be queried on keywords w that have been registered with the receiver, i.e. for which $H_1(w||v)$ is available, so the adversary cannot mount KGA on keyword guesses of its choice as both the trapdoor and dPEKS ciphertext inputs to

the dTest oracle would need to be functions of $H_1(w||v)$ corresponding to its keyword guess w. Table 1 shows the comparison between the schemes analyzed in this paper and the KG-sCKA dPEKS scheme in this section, with respect to the different security models.

7 Concluding remarks

We have proposed security models for both PEKS and dPEKS schemes in the sense of keyword guessing under chosen keyword attacks (KG-CKA). These models are more security sufficient than the KG-CKA model of [11] which is restrictive in the sense the adversary is not allowed any oracle query; so the latter adversarial capability is much weaker than the original adversarial capability considered by the IND-CKA model of Boneh et al. For the dPEKS setting, we focussed on the accessibility of the dTest function as an oracle, and based on this characterization then motivated the difference in the achieved security notions. This consideration corresponds well to the adversarial ability in the IND-CKA dPEKS notions of [12], and intuitively preserves the adversarial ability in moving from PEKS to dPEKS. We showed the applicability of these models in capturing KGA cryptanalysis of three recent dPEKS schemes of [10], [12] and [11]; such results cannot be captured by the weaker KG-CKA model of [11]. Since security models can be used in proving the security of a construction besides being used for cryptanalysis, a strong security model is desirable so that the corresponding proof guarantees security even against powerful adversaries; otherwise security collapses when an adversary exists in practice that has capabilities higher than those considered in the restricted (weaker) security model.

Acknowledgments We thank the anonymous reviewers for suggestions to emphasize the clear separation between the strengths of the different adversarial models, and to illustrate the feasibility of the strongest model i.e. KG-sCKA via a concrete scheme.

References

1. Abdalla M, Bellare M, Catalano D, Kiltz E, Kohno T, Lange T, Malone-Lee J, Neven G, Paillier P, Shi H (2005) Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions. In: Shoup V (ed) CRYPTO. Lecture Notes in computer science, vol 3621. Springer, pp 205–222
2. Baek J, Safavi-Naini R, Susilo W (2006) On the integration of public key data encryption and public key encryption with keyword search. In: Katsikas SK, Lopez J, Backes M, Gritzalis S, Preneel B (eds) ISC. Lecture Notes in Computer Science, vol 4176. Springer, pp 217–232
3. Baek J, Safavi-Naini R, Susilo W (2008) Public key encryption with keyword search revisited. In: Gervasi O, Murgante B, Laganà A, Taniar D, Mun Y, Gavrilova ML (eds) ICCSA (1). Lecture Notes in Computer Science, vol 5072. Springer, pp 1249–1259
4. Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G (2004) Public key encryption with keyword search. In: Cachin C, Camenisch J (eds) EUROCRYPT. Lecture Notes in Computer Science, vol 3027. Springer, pp. 506–522
5. Byun JW, Rhee HS, Park H-A, Lee DH (2006) Off-line keyword guessing attacks on recent keyword search schemes over encrypted data. In: Jonker W, Petkovic M (eds) Secure Data Management. Lecture Notes in Computer Science vol 4165. Springer, pp 75–83
6. Gu C, Zhu Y, Pan H (2007) Efficient public key encryption with keyword search schemes from pairings. In: Pei D, Yung M, Lin D, Wu C (eds) Inscrypt. Lecture Notes in Computer Science, vol 4990. Springer, pp 372–383
7. Jeong IR, Kwon JO, Hong D, Lee DH (2009) Constructing peks schemes secure against keyword guessing attacks is possible? Comput Commun 32(2):394–396
8. Park DJ, Kim K, Lee PJ (2004) Public key encryption with conjunctive field keyword search. In: Lim CH, Yung M (eds) WISA. Lecture Notes in Computer Science, vol 3325. Springer, pp 73–86
9. Rabinovich P (2010) A search engine for the global PKI. J Internet Serv Appl 1(2):83–93
10. Rhee HS, Park JH, Susilo W, Lee DH (2009) Improved searchable public key encryption with designated tester. Li W, Susilo W, Tupakula UK, Safavi-Naini R, Varadharajan V (eds) ASIACCS. ACM, New York, pp 76–379
11. Rhee HS, Park JH, Susilo W, Lee DH (2010) Trapdoor security in a searchable public-key encryption scheme with a designated tester. J Syst Softw 83(5):763–771
12. Rhee HS, Susilo W, Kim H-J (2009) Secure searchable public key encryption scheme against keyword guessing attacks. IEICE Electron Expr 6(5):237–243
13. Tang Q, Chen L (2009) Public-key encryption with registered keyword search. In: Martinelli F, Preneel B (eds) EuroPKI. Lecture Notes in Computer Science, vol 6391. Springer, pp 163–178
14. Yau W-C, Heng S-H, Goi B-M (2008) Off-line keyword guessing attacks on recent public key encryption with keyword search schemes. In: Rong C, Jaatun MG, Sandnes FE, Yang LT, Ma J (eds) ATC, Lecture Notes in Computer Science, vol 5060. Springer, pp 100–105