Keyword Updatable PEKS

Hyun Sook Rhee[1] and Dong Hoon Lee[2(✉)]

[1] Printing Solutions, Samsung Electronics Co. Ltd.,
Suwon-si, Gyeonggi-do, Korea
hyunsook.rhee@gmail.com
[2] Graduate School of Information Security, CIST,
Korea University, Seoul, Korea
donghlee@korea.ac.kr

Abstract. Secure keyword search in the asymmetric setting, also known as public-key encryption with keyword search (PEKS), enables a receiver to search the encrypted messages with a keyword without revealing any information on the messages to the server in the store-and-forward system such as an e-mail system. To make this possible, a sender encrypts a keyword with a receiver's public-key and tags the encrypted keyword to the messages. In the paper, we propose PEKS with keyword updatablility (KU-PEKS), where a tagged keyword can be updated upon the receiver's request. The proposed KU-PEKS is generically constructed and provides *ciphertext confidentiality* and *keyword-update privacy*. This keyword updatability enables synonym search and/or similarity search in PEKS. We also propose a generic transformation from KU-PEKS to secure keyword search in the symmetric setting, that is the first attempt to generically construct secure keyword search in the symmetric setting providing *trapdoor privacy*.

Keywords: Keyword updatable PEKS · Searchable encryption · Trapdoor privacy

1 Introduction

Secure keyword search enables a user to search the encrypted data with a keyword without revealing any information on the data. As increasing the concern of efficient searching techniques on massive encrypted data for providing data privacy against inside adversaries, secure keyword search has been extended to accommodate various queries such as equality, subset, range and inner product queries [1,6,8,9,11,17,19].

Secure keyword search can be classified into two types. (1) Secure keyword search in the asymmetric setting, also known as public-key encryption with keyword search (PEKS), was introduced by Boneh et al. [4], considering an application to the store-and-forward systems such as an e-mail system. In a PEKS, a sender generates a *ciphertext* CT_w of a keyword w under the public key of a receiver and sends the ciphertext CT_w along with an encrypted message to a server.

H. Kim and D. Choi (Eds.): WISA 2015, LNCS 9503, pp. 96–109, 2016.
DOI: 10.1007/978-3-319-31875-2_9

To retrieve the encrypted email messages containing a keyword w' from the server, a receiver asks to the server a query, that is a *trapdoor* $T_{w'}$ generated under the receiver's secret key. The server then runs a test function with inputs CT_w and $T_{w'}$ to decide whether or not $w = w'$, and forwards the corresponding email messages to the receiver. (2) In secure keyword search in the symmetric setting [5,17], the ciphertext CT_w of a keyword w is generated under a symmetric key and only the owner of the key can generate queries by using the symmetric key. Here a symmetric key is not shared, but owned by one client. The symmetric setting is suitable for a blog and web-hard service, where the same client uploads and downloads his/her data.

This paper firstly proposes a PEKS scheme with keyword updatability, called *keyword-updatable PEKS* scheme (*KU-PEKS* for short). Keyword updatability means that the server can update a ciphertext CT_w of keyword w into $\mathsf{CT}_{w'}$ of keyword w' by using a value, called keyword update value $k_{w \to w'}$ provided by a receiver in advance. This update process is different from re-encryption in the literature [2,10]. Keyword update process changes a corresponding plaintext of a ciphertext while re-encryption changes a person who can decrypt the ciphertext.

We can intuitionally and straightforwardly construct a direct construction from identity-based proxy reencryption (IBPRE) [10]. An IBPRE scheme is an identity-based one where a proxy can transform convert a ciphertext computed under Alice's ID into one that can be decrypted by Bob's secret key. In the direct construction, the server is provided with trapdoor T_w for finding CT_w and a keyword update key to transform CT_w into $\mathsf{CT}_{w'}$, where the key is derived from T_w. This implies that the server with T_w can derive any key to transform CT_w, say into $\mathsf{CT}_{w''}$, for any keyword w''. This is not intended by the receiver. To remedy this drawback, we construct an improved KU-PEKS scheme, where keyword update key can be generated only by the receiver. The improved scheme satisfies ciphertext confidentiality and consistency. In addition, KU-PEKS should satisfy the property, called keyword-update privacy which can update ciphertext without revealing any information on a keyword.

Application. One of the main applications of PEKS is an e-mail system, where a receiver retrieves e-mails with a keyword [4]. For example, the receiver may want to retrieve urgent e-mails with keyword "`urgent`". However it is highly possible that urgent e-mails in fact do not contain keyword "`urgent`". A sender might have selected a proper synonym as a keyword to express suitable situation. That is, urgent e-mail may contain "`urgency`", "`emergency`", or "`burning`" other than "`urgent`". It is certain that PEKS with enabling synonym search capability would be practically useful. This type of synonym search can be exploited for similarity search if we select a similar word with a few character difference instead of synonym. For example, "`urgent`" could be mistyped into `urgant`", "`urgenk`", and so on. Similarity search is not known to be possible in secure keyword search on encrypted data yet, while it is possible in web search engines. Similarity search in PEKS has been considered not attainable since one bit difference in a plaintext results in unexpected difference in the corresponding ciphertext.

One naive approach to enable similarity search in PEKS would be that a receiver tries each of all possible similar keywords, one by one. This inevitably delays retrieval time. Furthermore, more trapdoors available makes keyword guessing easier. Hence this is not a wise approach for similarity search. The generic construction to reduce retrieval time would be to provide the server with trapdoors for similar keywords in advance. For example, for keyword `urgent` a receiver provides ciphertext $\mathsf{CT}_{\texttt{urgent}}$ and similar trapdoors $T_{\texttt{urgency}}$, $T_{\texttt{emergency}}$, $T_{\texttt{burning}}$ and so on. When a ciphertext CT of a keyword similar to `urgent` is sent, the server runs the test function with inputs CT and each of the similar trapdoors. If the result of test function is `true`, the server replaces CT with $\mathsf{CT}_{\texttt{urgent}}$. This approach results in an identical ciphertext for similar keywords. That is, both $\mathsf{CT}_{\texttt{urgency}}$ and $\mathsf{CT}_{\texttt{emergency}}$ are transformed into $\mathsf{CT}_{\texttt{urgent}}$. However, this approach reveals the information that two original ciphertexts $\mathsf{CT}_{\texttt{urgency}}$ and $\mathsf{CT}_{\texttt{emergency}}$ encrypted two similar keywords even without keyword guessing. KU-PEKS construction makes the server transform CT into $\mathsf{CT}_{\texttt{urgent}}$ even without providing a trapdoor, where only the receiver can generate a keyword-update key. Furthermore, the transformed ciphertexts for similar keywords are random. That is, both $\mathsf{CT}_{\texttt{urgency}}$ and $\mathsf{CT}_{\texttt{emergency}}$ are transformed into independent random objects.

2 Preliminaries

We review the notation and the definitions of an identity-based proxy re-encryption (IBPRE) scheme. We assume that IBPRE satisfies (1) *unidirectionality* that IBPRE scheme permits a user U_1 to delegate to another user U_2 without permitting U_1 to decrypt U_2's ciphertexts and (2) *multiple-use capability* that IBPRE scheme permits the proxy to perform multiple consecutive re-encryptions on a ciphertext [2,10].

Notation: For any string x, $|x|$ denotes its length. For any set S, $|S|$ denotes its size. The symbol λ denotes a security parameter. We let $a \leftarrow b$ denote the assignment to a the result of evaluating b. We say a function μ is *negligible* if for any constant λ, there exists N such that $\mu(n) < 1/n^{\lambda}$ for $n > N$.

2.1 Identity-Based Proxy Re-encryption

An identity-based proxy re-encryption (IBPRE) scheme is an extension of identity based encryption (IBE) scheme in which the decryption capability of a user to another user via a third party called a *proxy*. Via $(\mathsf{PP}, \mathsf{msk}) \leftarrow \mathbf{Setup}(1^{\lambda})$ the setup algorithm produces a pair of public parameters and master secret key for security parameter λ; via $sk_{id} \leftarrow \mathbf{KeyGen}(\mathsf{PP}, \mathsf{msk}, id)$ the key generation algorithm takes the public parameters PP and the master secret key msk and an identity $id \in \{0,1\}^*$ as input and produces a secret key sk_{id} corresponding to that identity id; via $c_{id} \leftarrow \mathbf{Enc}(\mathsf{PP}, id, m)$ the encryption algorithm encrypts a message m to create a ciphertext c_{id} under the specified identity; via $rk_{id_1 \rightarrow id_2} \leftarrow \mathbf{RKGen}(\mathsf{PP}, sk_{id_1}, id_1, id_2)$ the re-encryption key generation

algorithm generates a re-encryption key $rk_{id_1 \to id_2}$ for converting a ciphertext for one identity id_1 into for the other identity id_2, which are given to the proxy; via $c_{id_2} \leftarrow$ **Reencrypt**(PP, $rk_{id_1 \to id_2}$, c_{id_1}) the re-encryption algorithm reencrypts a ciphertext c_{id_1} to produce a "re-encrypted"ciphertext c_{id_2}; via $m \leftarrow$ **Decrypt**(PP, sk_{id}, c_{id}) the decryption algorithm decrypts a ciphertext c_{id} to recover a message m.

Correctness of IBPRE: In [10], Green and Ateniese identified the concept of the correctness of IBPRE schemes. Suppose that $(PP, msk) \leftarrow$ **Setup**(1^λ), $d_{id} \leftarrow$ **KeyGen**(PP, msk, id). Let id and id_1 be any identities. Suppose that $rk_{id_1 \to id} \leftarrow$ **RKGen**(PP, sk_{id_1}, id_1, id) and $C_{id_1} \leftarrow$ **Enc**(PP, id_1, m). If $C_{id} \leftarrow$ **Enc**(PP, id, m) or $C_{id} \leftarrow$ **Reencrypt**(PP, $rk_{id_1 \to id}$, C_{id_1}), then the following propositions hold:

- **Decrypt**(PP, sk_{id_1}, C_{id_1}) $= m$
- **Decrypt**(PP, sk_{id}, **Reencrypt**(PP, $rk_{id_1 \to id}$,C_{id_1})) $= m$

Game$_1$: The game for *ciphertext confidentiality* [10] between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ proceeds as follows.

- **Setup**: $\mathcal{B}$ runs the **Setup**(1^λ) algorithm to obtain ($\mathsf{PP}, \mathsf{msk}$) and sends the public parameter PP to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ makes the following queries and $\mathcal{B}$ adaptively responses as follows.
  - For any extraction query id, $\mathcal{B}$ returns $d_{id} =$**KeyGen**(PP, msk, id) to $\mathcal{A}$.
  - For any reencryption key query (id_1, id_2), $\mathcal{B}$ extracts the key $d_{id_1} =$**KeyGen** (PP, msk, id_1) and returns $rk_{id_1 \to id_2} =$**RKGen**(PP, sk_{id_1}, id_1, id_2).
- **Challenge**: $\mathcal{A}$ sends identity id^* and messages m_0 and m_1 to the challenger. $\mathcal{B}$ picks a random $\beta \in \{0, 1\}$ and returns $c_\beta \leftarrow$ **Enc**($\mathsf{PP}, id^*, m_\beta$) to $\mathcal{A}$. The restriction is as follows.
  - The extraction query id^* and the re-encryption key query (id^*, id') and the extraction query id' should not be queried previously, for any identity id'.
- **Phase 2**: $\mathcal{A}$ makes the following queries as in the **Phase 1**, except for the following queries.
  - The extraction query id^* and the re-encryption key query (id^*, id') and the extraction query id' should not be queried previously, for any identity id'.
- **Guess**: The adversary returns a guess $\beta' \in \{0, 1\}$ of β.

The advantage of $\mathcal{A}$ in breaking IND-CPA security in an **IBPRE** scheme is defined as

$$\mathbf{Adv}^{\mathsf{ibpre\text{-}ind\text{-}cpa}}_{\mathbf{IBPRE},\mathcal{A}}(\lambda) = |\Pr[\beta = \beta'] - 1/2| \ .$$

Definition 1. *We say that **IBPRE** satisfies a ciphertext confidentiality (IND-CPA-secure) if the advantage* $\boldsymbol{Adv}^{\mathsf{ibpre\text{-}ind\text{-}cpa}}_{\boldsymbol{IBPRE},\mathcal{A}}(\lambda)$ *of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ is negligible in the security parameter λ.*

Game$_2$: The game for *anonymity of IBPRE* between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ proceeds as follows.

- **Setup**: $\mathcal{B}$ runs the **Setup**(1^λ) algorithm to obtain $(\mathsf{PP}, \mathsf{msk})$ and sends the public parameter PP to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ makes the following queries and $\mathcal{B}$ adaptively responses as follows.
  - For any extraction query id, $\mathcal{B}$ returns d_{id} =**KeyGen**(PP, msk, id) to $\mathcal{A}$.
  - For any re-encryption key query (id_1, id_2), $\mathcal{B}$ extracts the key d_{id_1}=**KeyGen** (PP, msk, id_1) and returns $rk_{id_1 \to id_2}$ =**RKGen**(PP, sk_{id_1}, id_1, id_2).
- **Challenge**: $\mathcal{A}$ sends identities id_0^* and id_1^* and a message m^* to the challenger, and $id_0^* \neq id_1^*$. $\mathcal{B}$ picks a random $\beta \in \{0,1\}$ and returns $C_\beta \leftarrow$ **Enc**$(\mathsf{PP}, id_\beta^*, m^*)$ to $\mathcal{A}$. The restriction is that $\mathcal{A}$ does not hold a secret keys $d_{id_0^*}$ and $d_{id_1^*}$, and reencryption keys $rk_{id_0^* \to id}$, $rk_{id_1^* \to id}$ and a secret key d_{id}, for any identity id.
- **Phase 2**: $\mathcal{A}$ makes the following queries as in the **Phase 1**, except for the following queries.
  - The extraction query id_1^* and re-encryption key extraction queries (id_1^*, id') and the extraction query id' should be restricted, for any id'.
- **Guess**: The adversary returns a guess $\beta' \in \{0,1\}$ of β.

The advantage of $\mathcal{A}$ in breaking ANO-CPA security in an **IBPRE** scheme is defined as

$$\mathbf{Adv}^{\text{ibpre-ano-cpa}}_{\mathbf{IBPRE},\mathcal{A}}(\lambda) = |\Pr[\beta = \beta'] - 1/2| \ .$$

Definition 2. *We say that **IBPRE** satisfies the anonymity (ANO-CPA-secure) if the advantage* $\boldsymbol{Adv}^{\text{ibpre-ano-cpa}}_{\boldsymbol{IBPRE},\mathcal{A}}(\lambda)$ *of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ is negligible in the security parameter λ.*

Remark 1. The notion of *key-privacy* or *anonymity* [3] in public-key encryption schemes has been considered as the security requirements of encryption schemes as well as *semantic security*. Ateniese et al. introduced a notion of key privacy of proxy re-encryption scheme [2]. For anonymity in **IBPRE**, an adversary cannot distinguish the intended recipient from the ciphertexts and re-encryption keys even when given re-encryption keys and re-encryption oracles. As mentioned in [2], the anonymity of ciphertext (Definition 3) as well as the re-encryption key privacy (Definition 4) should be considered in **IBPRE**.

We can infer that the IBPRE scheme proposed by Green and Ateniese [10] satisfies the anonymity (IBPRE-ANO-CPA security) against chosen-plaintext attacks.

Re-encryption key privacy of IBPRE: We first define the anonymity (IBPRE-REKey-CPA security) of the IBPRE scheme against adaptive chosen-plaintext attacks using **Game**$_3$.

Game$_3$: The game for *re-encryption predicate privacy* between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ proceeds as follows.

- **Setup**: $\mathcal{B}$ runs the **Setup**(1^λ) algorithm to obtain $(\mathsf{PP}, \mathsf{msk})$ and sends the public parameter PP to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ makes the following queries and $\mathcal{B}$ adaptively responses as follows.

- For any extraction query id, $\mathcal{B}$ returns d_{id} =**KeyGen**(PP, msk, id) to $\mathcal{A}$.
- For any reencryption key query (id_1, id_2), $\mathcal{B}$ extracts the key d_{id_1}=**KeyGen** (PP, msk, id_1) and returns $rk_{id_1 \to id_2}$ =**RKGen**(PP, sk_{id_1}, id_1, id_2).

– **Challenge**: $\mathcal{A}$ sends identities id_0^* and id_1^* and a message m^* to the challenger, and $id_0^* \neq id_1^*$. $\mathcal{B}$ computes $s \leftarrow$ **RKGen**($\mathsf{PP}, sk_{id_0^*}, id_0^*, id_1^*$) and picks a random $\beta \in \{0,1\}$. $\mathcal{B}$ returns $c_\beta = s$ to $\mathcal{A}$ if $\beta = 1$ and a random key in the key space otherwise. The restriction is that $\mathcal{A}$ does not hold secret keys $d_{id_0^*}$ and $d_{id_1^*}$, and a reencryption key $rk_{id_0^* \to id_1^*}$.
– **Phase 2**: $\mathcal{A}$ makes the following queries as in the **Phase 1**, except for the following queries.
  - The extraction query id_1^* should be restricted.
  - The re-encryption key extraction queries (id_1^*, id') and the extraction query id' should be restricted, for any id'.
– **Guess**: The adversary returns a guess $\beta' \in \{0,1\}$ of β.

The advantage of $\mathcal{A}$ in breaking REKey-CPA security in an **IBPRE** scheme is defined as

$$\mathbf{Adv}^{\mathsf{ibpre\text{-}rekey\text{-}cpa}}_{\mathbf{IBPRE},\mathcal{A}}(\lambda) = |\Pr[\beta = \beta'] - 1/2| \ .$$

Definition 3. *We say that **IBPRE** satisfies a privacy of re-encryption key (REKey-CPA-secure) if the advantage* $\boldsymbol{Adv}^{\mathsf{ibpre\text{-}rekey\text{-}cpa}}_{\boldsymbol{IBPRE},\mathcal{A}}(\lambda)$ *of any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ is negligible in the security parameter λ.*

3 Keyword-Updatable PEKS

Definition 4. *(Keyword Updatable PEKS Scheme) A keyword-updatable PEKS scheme (**KU-PEKS**) consists of six PPT algorithms as follows.*

- **Gen**(λ) takes a security parameter λ as input, and generates a pair of public and secret keys (PK, SK) of the receiver R.
- **Td**(SK, w) takes as inputs a receiver's secret key SK and a keyword, w. It then generates a trapdoor T_w.
- **PEKS**(PK, w) takes as inputs a receiver's public key PK and a keyword, w. It returns a ciphertext CT on the keyword w.
- **Test**(CT, T_w) takes as inputs a ciphertext CT and a trapdoor T_w. It outputs '1' if $w = w'$ and '0' otherwise, where $\mathsf{CT} =$ **PEKS**(PK, w').
- **kuTd**(PK, T_{w_1}, w_1, w_2) takes as inputs a receiver's public key PK, a trapdoor T_{w_1}, and keywords w_1 and w_2. It then outputs a keyword-update value $\mathsf{kuTd}_{w_1 \to w_2}$.
- **kuPEKS**($PK, \mathsf{CT}, \mathsf{kuTd}_{w_1 \to w_2}$) takes as inputs a receiver's public key PK, a ciphertext $\mathsf{CT} =$ **PEKS**(PK, w), and a keyword-update value $\mathsf{kuTd}_{w_1 \to w_2} =$ **kuTd**(PK, T_{w_1}, w_1, w_2). It outputs $\mathsf{CT}' =$ **PEKS**(PK, w_2) if $w = w_1$ and aborts otherwise.

Table 1. Consistency of KU-PEKS.

$\mathbf{Exp}^{\text{kupeks-cons}}_{\mathbf{kuPEKS},\mathcal{A}}(\lambda)$
$(PK, SK) \leftarrow \mathbf{Gen}(\lambda)$; $(w, w') \leftarrow \mathcal{A}(PK)$; $\mathsf{CT} \leftarrow \mathbf{PEKS}(PK, w)$
$\mathsf{kuTd}_{w_1 \to w_2} \leftarrow \mathbf{kuTd}(SK, PK, T_{w_1}, w_1, w_2)$; $T_{w'} \leftarrow \mathbf{Td}(SK, w')$
$\mathsf{CT}' \leftarrow \mathbf{kuPEKS}(PK, \mathsf{CT}, \mathsf{kuTd}_{w_1 \to w_2})$
If $w \neq w_1$ or $w_2 \neq w'$ and $\mathbf{Test}(T_{w'}, \mathsf{CT}') = 1$ then return 1 else return 0

Suppose there exists an adversary $\mathcal{A}$ that wants to make consistency fail. A computational consistency for KU-PEKS scheme is defined as follows (Table 1). The advantage of $\mathcal{A}$ is defined as follows.

$$\mathbf{Adv}^{\text{kupeks-cons}}_{\mathbf{kuPEKS},\mathcal{A}}(\lambda) = \Pr[\mathbf{Exp}^{\text{kupeks-cons}}_{\mathbf{kuPEKS},\mathcal{A}}(\lambda) = 1],$$

where the probability is taken over all possible coin flips of all the algorithms involved.

Definition 5. *We say that **KU-PEKS** satisfies "computationally consistency" if for any PPT adversary $\mathcal{A}$ attacking **KU-PEKS** scheme the advantage $\boldsymbol{Adv}^{\text{kupeks-cons}}_{\boldsymbol{kuPEKS},\mathcal{A}}(\lambda)$ is negligible.*

Game$_4$: The game for *ciphertext confidentiality* between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ proceeds as follows.

- **Setup**: $\mathcal{B}$ runs the $\mathbf{Gen}(1^\lambda)$ algorithm to obtain (PK, SK) and sends the public key PK to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ makes the following queries and $\mathcal{B}$ adaptively responses as follows.
  - **Trapdoor queries**: For any trapdoor query of the form $w \in \{0,1\}^*$, $\mathcal{B}$ returns T_w=$\mathbf{Td}(SK, w)$ to $\mathcal{A}$.
  - **Keyword-Update Key queries**: For any keyword-update key query (w_1, w_2), $\mathcal{B}$ returns $\mathsf{kuTd}_{w_1 \to w_2}$ =$\mathbf{kuTd}(PK, T_{w_1}, w_1, w_2)$.
- **Challenge**: $\mathcal{A}$ sends keywords w_0^* and w_1^* to the challenger. The restriction is that $\mathcal{A}$ did not previously make both (1) the trapdoor query of w_b^* and (2) the keyword-update value $\mathsf{kuTd}_{w_b^* \to w}$ as well as the trapdoor query T_w, for any keyword w and $b = 0, 1$. $\mathcal{B}$ picks a random $\beta \in \{0,1\}$ and returns the challenge ciphertext $\mathsf{CT}^*_\beta \leftarrow \mathbf{PEKS}(PK, w^*_\beta)$.
- **Phase 2**: $\mathcal{A}$ makes the following queries as in the **Phase 1**, except for the following queries.
  - The trapdoor query of the form $w_0{}^*$ and $w_1{}^*$ should be restricted.
  - The keyword-update key queries of the form $(w_b{}^*, w')$ and the trapdoor query of the form w' should be not queried, for any keyword $w' \in \{0,1\}^*$ and $b = 0, 1$.
- **Guess**: The adversary returns a guess $\beta' \in \{0,1\}$ of β.

The advantage of $\mathcal{A}$ in breaking the ciphertext confidentiality (kuPEKS-IND-CPA security) in **KU-PEKS** scheme is defined as

$$\mathbf{Adv}^{\text{kupeks-ind-cpa}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(\lambda) = |\Pr[\beta = \beta'] - 1/2| \ .$$

Definition 6. *We say that a KU-PEKS scheme provides the ciphertext confidentiality against an adaptive chosen-plaintext attack (kuPEKS-IND-CPA security) if for any PPT adversary $\mathcal{A}$, $\boldsymbol{Adv}^{\textit{kupeks-ind-cpa}}_{\boldsymbol{kuPEKS},\,\mathcal{A}}(\lambda)$ is negligible.*

Game$_5$: The game for *keyword-update key privacy* between the adversary $\mathcal{A}$ and the challenger $\mathcal{B}$ proceeds as follows.

- **Setup**: $\mathcal{B}$ runs the $\mathbf{Gen}(1^\lambda)$ algorithm to obtain (PK, SK) and sends the public key PK to $\mathcal{A}$.
- **Phase 1**: $\mathcal{A}$ makes the following queries and $\mathcal{B}$ adaptively responses as follows.
  - **Trapdoor queries**: For any trapdoor query of the form $w \in \{0,1\}^*$, $\mathcal{B}$ returns T_w=$\mathbf{Td}(SK, w)$ to $\mathcal{A}$.
  - **Keyword-Update Key queries**: For any keyword-update key query (w_1, w_2), $\mathcal{B}$ returns $\mathsf{kuTd}_{w_1 \to w_2}$ =$\mathbf{kuTd}(PK, T_{w_1}, w_1, w_2)$.
- **Challenge**: $\mathcal{A}$ queries w to obtain $T_w \leftarrow \mathbf{Td}(SK, w)$, for any keyword $w \neq w_b^*$ $(b = 0, 1)$ and sends T_w as well as keywords w_0^* and w_1^* to the challenger $\mathcal{B}$. The restriction is that $\mathcal{A}$ did not previously make both (1) the trapdoor query of w_b^* and (2) the keyword-update key $\mathsf{kuTd}_{w_b^* \to w}$ as well as the trapdoor query T_w, for any keyword w and $b = 0, 1$. $\mathcal{B}$ picks a random $\beta \in \{0,1\}$ and returns the challenge keyword-update key $\mathsf{kuTd}_{w \to w_\beta^*} \leftarrow \mathbf{kuTd}\ (PK, T_w, w, w_\beta^*)$ to $\mathcal{A}$.
- **Phase 2**: $\mathcal{A}$ makes the following queries as in the **Phase 1**, except for the following queries.
  - The trapdoor query of the form $w_0{}^*$ and $w_1{}^*$ should be restricted.
  - The keyword-update key queries of the form $(w_b{}^*, w')$ and the trapdoor query of the form w' should be not queried, for any keyword $w' \in \{0,1\}^*$ and $b = 0, 1$.
- **Guess**: The adversary returns a guess $\beta' \in \{0,1\}$ of β.

The advantage of $\mathcal{A}$ in breaking the keyword-update key privacy (kuTd-IND-CPA security) in **KU-PEKS** scheme is defined as

$$\mathbf{Adv}^{\mathsf{kuTd\text{-}ind\text{-}cpa}}_{\mathbf{KU\text{-}PEKS},\,\mathcal{A}}(\lambda) = |\Pr[\beta = \beta'] - 1/2| \ .$$

Definition 7. *We say that KU-PEKS scheme provides the keyword-update key privacy against an adaptive chosen-plaintext attack (kuTd-IND-CPA security) if for any PPT adversary $\mathcal{A}$, $\boldsymbol{Adv}^{\textit{kuTd-ind-cpa}}_{\boldsymbol{kuPEKS},\,\mathcal{A}}(\lambda)$ is negligible.*

4 Construction of KU-PEKS

In an IBE scheme $\mathcal{IBE} = (\mathbf{Setup}_{\mathcal{IBE}}, \mathbf{KeyDer}_{\mathcal{IBE}}, \mathbf{Enc}_{\mathcal{IBE}}, \mathbf{Dec}_{\mathcal{IBE}})$, the key generator with master public/private key pair (PK, SK) generates the secret key sk_{ID} corresponding to an identity ID by computing $\mathbf{KeyDer}_{\mathcal{IBE}}\ (SK, \mathsf{ID})$.

We now propose a generic construction *keyword updatability* PEKS from IBPRE. We can intuitionally and straightforwardly construct a direct construction from IBPRE along the line of the approach in [1]. The idea of providing keyword-updatability is that re-encryption key $rk_{w \to w'}$ that converts C_w into

$C_{w'}$ without changing the message in IBPRE is used in generating keyword-update key $\mathsf{k}_{w\to w'}$ for converting ciphertext CT_w with ciphertext $\mathsf{CT}_{w'}$ in KU-PEKS. Unfortunately, the direct construction inherits the property from IBPRE that anyone with secret key sk_w can generate re-encryption key $rk_{w\to w''}$, for any identity w'', by the re-encryption key generation algorithm. Moreover, it is required that anyone with re-encryption key $rk_{w\to w'}$ and secret key $sk_{w'}$ can generate re-encryption key $rk_{w\to w''}$, for any identity w'' that is called collusion attacks of a proxy and a delegator [16]. This property in KU-PEKS means that anyone who obtains trapdoor $T_w(= sk_w)$ or keyword-update key $\mathsf{k}_{w\to w'} = [T_w, rk_{w\to w'}]$ can generate new keyword-update key $\mathsf{k}_{w\to w''} = [sk_w, rk_{w\to w''}]$ for any keyword w''. To overcome this weakness, the underlying IBPRE needs to be secure against collusion attacks. Koo et al.'s collusion-resistant IBPRE scheme [16] can be a proper candidate for the underlying IBPRE.

A generic approach transforms an IBPRE scheme secure against collusion attacks [16] into a secure KU-PEKS scheme. To perform the keyword update, the server first needs to find the PEKS ciphertexts that a receiver wants to retrieve. For this purpose, the server in the first approach is given a trapdoor information together with a keyword-update key. The novel idea of the second approach is that the server is enabled to find the corresponding PEKS ciphertexts without any trapdoor information. Let $rk_{w_1\to w_2}$ be a re-encryption key in IBPRE. The keyword-update key $\mathsf{k}_{w_1\to w_2}$ of the transformed KU-PEKS consists of two re-encryption keys $rk_{w_1\to rand}$ and $rk_{w_1\to w_2}$ and secret key sk_{rand} for random $rand$. Here, $rk_{w_1\to rand}$ and sk_{rand} are used in searching a ciphertext CT_{w_1}, and $rk_{w_1\to w_2}$ is used in converting CT_{w_1} into CT_{w_2}. As stated above, if the underlying IBPRE scheme is secure against collusion attacks, anyone with $rk_{w_1\to rand}$, sk_{rand} and $rk_{w_1\to w_2}$ cannot obtain sk_{w_1} or $rk_{w_1\to w_2'}$ for some keyword $w_2'(\neq w_2)$ from the given information.

The generic construction of KU-PEKS scheme **KU-PEKS** =(**Gen, Td, PEKS, kuTd, kuPEKS, Test**), using **IBPRE**=(**Setup, KeyGen, Enc, RKGen, Reencrypt, Decrypt**), proceeds as follows.

- **Gen**(1^λ): This algorithm runs **Setup**(1^λ) to obtain $(\mathsf{PP}, \mathsf{msk})$. The public key is $PK = \mathsf{PP}$ and the secret key is $SK = \mathsf{msk}$. It outputs $(PK, SK) = (\mathsf{PP}, \mathsf{msk})$.
- **Td**(SK, w)**:** Let $w \in \{0,1\}^n$ be a keyword. To generate a trapdoor T_w of w, the trapdoor algorithm runs $d_w \leftarrow$ **KeyGen**$(\mathsf{PP}, \mathsf{msk}, w)$. The trapdoor is $T_w = d_w$.
- **PEKS**(PK, w): To encrypt a keyword $w \in \{0,1\}^*$ under the public key PK, this algorithm picks a random message $R \in \mathcal{M}$ and computes $C \leftarrow$ **Enc**(PK, w, R) and outputs $\mathsf{CT} = [C_1, C_2] = [R,\ C]$.
- **Test**$(PK, \mathsf{CT}, T_{w'})$ **:** To obtain the test result, this algorithm parses the ciphertext CT as $[C_1, C_2]$. It computes $R' \leftarrow$ **Decrypt**$(PK, T_{w'}, C_2)$. It outputs '0' if $C_1 \neq R'$ and '1' otherwise.
- **kuTd**(PK, T_{w_1}, w_1, w_2) **:** To generate a keyword-update key from w_1 to w_2, this algorithm chooses a random identity $id \in \{0,1\}^*$ and computes $d_{id} \leftarrow$ **KeyGen**$(\mathsf{PP}, \mathsf{msk}, id)$. It computes $rk_{w_1\to w_2} \leftarrow$ **RKGen**$(PK,\ T_{w_1},\ w_1,$

w_2) and $rk_{w_1 \to id} \leftarrow$**RKGen**$(PK,\ T_{w_1},\ w_1,\ id)$ and outputs $\mathsf{kuTd}_{w_1 \to w_2} = [rk_{w_1 \to w_2}, rk_{w_1 \to id}, T_{id}]$.

– **kuPEKS**$(PK, \mathsf{CT}, \mathsf{kuTd}_{w_1 \to w_2})$: Let $\mathsf{kuTd}_{w_1 \to w_2} = [K_1, K_2, K_3]$ be a keyword-update key and $\mathsf{CT} = [C_1, C_2]$ be a ciphertext. This algorithm proceeds as follows.
  - It computes $C_2' \leftarrow$ **Reencrypt**(PK, K_1, C_2) and $C_2'' \leftarrow$ **Reencrypt** (PK, K_2, C_2).
  - It aborts if **Decrypt**$(PK, K_3, C_2'') \neq C_1$ and outputs $\mathsf{CT}_{w_2} = [C_1, C_2']$ otherwise.

We now prove that our generic construction provides the ciphertext confidentiality, the keyword-update key privacy and the computational consistency as follows. We note that this proof approach follows from [1].

Theorem 1. *If the underlying IBE scheme of **IBPRE** scheme provides IBE-ANO-CPA security, then our generic construction provides the kuPEKS-IND-CPA security.*

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ attacking the kuPEKS-IND-CPA security of KU-PEKS scheme. We can construct a PPT adversary $\mathcal{B}$ attacking the IBE-ANO-CPA security of the underlying IBE scheme of IBPRE scheme. Let $\mathcal{C}$ denote a challenger against $\mathcal{B}$. $\mathcal{C}$ begins by supplying $\mathcal{B}$ with the public parameters PP of **IBPRE** and $\mathcal{B}$ forwards PP (as the public key PK of KU-PEKS) to $\mathcal{A}$.

In its `find` stage, given public key PP, $\mathcal{A}$ runs $\mathcal{B}(\mathtt{find}, \mathsf{PP})$ to obtain challenge keywords w_0^* and w_1^*. $\mathcal{B}$ chooses a random message $R^* \in \mathcal{M}$ and gives a challenge query (w_0^*, w_1^*, R^*) to $\mathcal{C}$. $\mathcal{B}$ mounts an IBPRE-ANO-CPA attack on **IBPRE** by interacting with $\mathcal{A}$ as follows.

- On trapdoor query w, $\mathcal{B}$ makes an extraction query with an identity w to $\mathcal{C}$. Upon receiving w, $\mathcal{C}$ runs the KeyGen algorithm in **IBPRE** to obtain the private key d_w. $\mathcal{C}$ returns the trapdoor $d_w (= T_w)$.
- On keyword-update key query (w, w'), $\mathcal{B}$ makes a re-encryption key query with an identity (w, w') to $\mathcal{C}$. Upon receiving (w, w'), $\mathcal{C}$ runs the RKGen algorithm in **IBPRE** to obtain the re-encryption key $rk_{w \to w'}$. $\mathcal{C}$ returns the keyword-update key $rk_{w \to w'} (= \mathsf{kuTd}_{w_1 \to w_2})$.

On receiving $C_b^* =$ **Enc**$(\mathsf{PP}, w_b^*, R^*)$ from $\mathcal{C}$, $\mathcal{B}$ gives back his challenge ciphertext $[R^*, C_b^*]$ to $\mathcal{A}$ in its `guess` stage. It is easy to see from the definition of the KU-PEKS scheme that CT^* is equal to the output of the encryption algorithm in **KU-PEKS** for the input (PP, w_b^*). For any keyword $w \in \{0,1\}^*$, we insist that if $\mathcal{B}$ makes a trapdoor query on w_b^* $(b = 0, 1)$ or a keyword-update key $rk_{w_b \to w}$ on (w_b, w) in some phases, then $\mathcal{B}$ aborts. Eventually, $\mathcal{A}$ must guess b' for b. Then $\mathcal{B}$ outputs b' as its guess for b. It is easy to see that for any b $(\in \{0,1\})$,

$$\Pr[\mathbf{Exp}^{\mathsf{ibe\text{-}ano\text{-}cpa\text{-}b}}_{\mathbf{IBPRE},\mathcal{B}}(k) = 1] = \Pr[\mathbf{Exp}^{\mathsf{kupeks\text{-}ind\text{-}cpa\text{-}b}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(k) = 1].$$

Therefore, $\mathbf{Adv}^{\mathsf{kupeks\text{-}ind\text{-}cpa}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\mathsf{ibe\text{-}ano\text{-}cpa}}_{\mathbf{IBPRE},\mathcal{B}}(k)$.

Theorem 2. *If **IBPRE** scheme provides the IBPRE-ANO-CPA security, then our generic construction provides the kuTd-IND-CPA security.*

Proof. Suppose that there exists a PPT adversary $\mathcal{A}$ attacking the kuTd-IND-CPA security of KU-PEKS scheme. We can construct a PPT adversary $\mathcal{B}$ attacking the IBPRE-ANO-CPA security of IBPRE scheme. Let $\mathcal{C}$ denote a challenger against $\mathcal{B}$. $\mathcal{C}$ begins by supplying $\mathcal{B}$ with the public parameters PP of **IBPRE** and $\mathcal{B}$ forwards PP (as the public key PK of KU-PEKS) to $\mathcal{A}$.

In its `find` stage, given public key PP, $\mathcal{A}$ runs $\mathcal{B}(\mathtt{find}, \mathsf{PP})$ to obtain challenge keywords w_0^* and w_1^*. $\mathcal{B}$ chooses a random message $R^* \in \mathcal{M}$ and keyword w_1 and computes $\mathsf{CT}_{w_1} \leftarrow \mathbf{Enc}(\mathsf{PP}, w_1, R^*)$ and queries w_1 to the private-key generation oracle to obtain $d_{w_1} \leftarrow \mathbf{KeyGen}(\mathsf{PP}, \mathsf{msk}, w_1)$. $\mathcal{B}$ gives a challenge query $(T_{w_1}, w_1, w_0^*, w_1^*)$ to $\mathcal{C}$. $\mathcal{B}$ mounts an IBPRE-ANO-CPA attack on **IBPRE** by interacting with $\mathcal{A}$ as follows.

- On trapdoor query w, $\mathcal{B}$ makes an extraction query with an identity w to $\mathcal{C}$. Upon receiving w, $\mathcal{C}$ runs the KeyGen algorithm in **IBPRE** to obtain the private key d_w. $\mathcal{C}$ returns the trapdoor $d_w (= T_w)$.
- On keyword-update key query (w, w'), $\mathcal{B}$ makes a re-encryption key query with an identity (w, w') to $\mathcal{C}$. Upon receiving (w, w'), $\mathcal{C}$ runs the RKGen algorithm in **IBPRE** to obtain the re-encryption key $rk_{w \to w'}$. $\mathcal{C}$ returns the keyword-update key $rk_{w \to w'} (= \mathsf{kuTd}_{w_1 \to w_2})$.

On receiving $rk_{w_1 \to w_b^*} = \mathbf{RKGen}(\mathsf{PP}, d_{w_1}, w_b^*)$ from $\mathcal{C}$, $\mathcal{B}$ randomly chooses $id \in \{0,1\}^*$ and runs the private-key generation oracle to obtain T_{id} and computes $rk_{w_1 \to id}$ with d_{w_1}. $\mathcal{B}$ gives back his challenge keyword update key $kuTd_{w_1 \to w_b^*} = [rk_{w_1 \to w_b^*},\ rk_{w_1 \to id}, T_{id}]$ to $\mathcal{A}$ in its `guess` stage. It is easy to see from the definition of the KU-PEKS scheme that $kuTd_{w_1 \to w_b^*}$ is equal to the output of the keyword update key generation algorithm in **KU-PEKS** for the input $(PK, T_{w_1}, w_1, w_b^*)$.

For any keyword $w \in \{0,1\}^*$, we insist that if $\mathcal{B}$ makes a trapdoor query on w_b^* $(b = 0, 1)$ or a keyword-update key $rk_{w_b \to w}$ on (w_b, w) in some phases, then $\mathcal{B}$ aborts. Eventually, $\mathcal{A}$ must guess b' for b. Then $\mathcal{B}$ outputs b' as its guess for b. It is easy to see that for any b $(\in \{0,1\})$,

$$\Pr[\mathbf{Exp}^{\mathsf{ibpre\text{-}ano\text{-}cpa\text{-}b}}_{\mathbf{IBPRE},\mathcal{B}}(k) = 1] = \Pr[\mathbf{Exp}^{\mathsf{kuTd\text{-}ind\text{-}cpa\text{-}b}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(k) = 1].$$

Therefore, $\mathbf{Adv}^{\mathsf{kuTd\text{-}ind\text{-}cpa}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\mathsf{ibpre\text{-}ano\text{-}cpa}}_{\mathbf{IBPRE},\mathcal{B}}(k)$.

Theorem 3. *If **IBPRE** scheme satisfies the correctness, then our construction is computationally consistent.*

Proof. Suppose that there is a PPT adversary $\mathcal{A}$ attacking the computational consistency of KU-PEKS scheme **KU-PEKS**. We can construct a PPT adversary $\mathcal{B}$ attacking the IBPRE-IND-CPA security of IBPRE scheme **IBPRE**.

To show the computational consistency of **IBPRE**, we will show that if $\mathcal{A}$ successes to attack the computational consistency of **KU-PEKS** then $\mathcal{B}$ successes to attack the IBPRE-IND-CPA security of **IBPRE** or it contradicts the correctness constraint of **IBPRE**. We can construct algorithms $\mathcal{B}$ as follows.

Let $\mathcal{C}$ denote a challenger against $\mathcal{B}$. $\mathcal{C}$ begins by supplying $\mathcal{B}$ with the public parameters PP of **IBPRE**. Given public parameters $\mathsf{PP} = PK$, $\mathcal{B}$ runs $\mathcal{A}(\mathsf{PP})$ to obtain keywords w and w' such that $\mathbf{Test}(\mathsf{CT}_w, T_{w'}) = 1$, where $\mathsf{CT}_w = \mathbf{PEKS}(\mathsf{PP}, w)$, (or $\mathsf{CT}_w = \mathsf{kuPEKS}(\mathsf{PP}, T_{w_0}, w_0, w)$, for some $w_0 \in \{0,1\}^*$) and $T_{w'} = \mathbf{Td}(\mathsf{msk}, w')$.

In simulation, $\mathcal{B}$ will eventually obtain in one of two different ways:

Case I $\mathcal{A}$ will obtain keywords w and w' such that $\mathbf{Test}(\mathsf{CT}_w, T_{w'}) = 1$, where $\mathsf{CT}_w = \mathbf{PEKS}(\mathsf{PP}, w)$ and $T_{w'} = \mathbf{Td}(\mathsf{msk}, w')$. In this case, $\mathcal{B}$ needs to generate all secret keys, except the key d_w.

Case II $\mathcal{A}$ will obtain keywords w and w' such that $\mathbf{Test}(\mathsf{CT}_w, T_{w'}) = 1$, where $\mathsf{kuTd}_{w_0 \to w} = \mathbf{kuTd}(PK, T_{w_0}, w_0, w)$, $\mathsf{CT}_w = \mathsf{kuPEKS}(\mathsf{PP}, \mathsf{CT}_{w_0}, \mathsf{kuTd}_{w_0 \to w})$ and $T_{w_0} = \mathbf{Td}(\mathsf{msk}, w_0)$ (for some $w_0 \in \{0,1\}^*$) and $T_{w'} = \mathbf{Td}(\mathsf{msk}, w')$. In this case, $\mathcal{B}$ needs to generate all secret keys, except the key d_w and d_{w_0}.

$\mathcal{B}$ chooses random messages $R_0^*, R_1^* \in \mathcal{M}$ and gives a challenge query (w, R_0^*, R_1^*) to $\mathcal{C}$. $\mathcal{B}$ gets back a ciphertext $C^* = \mathbf{Enc}(\mathsf{PP}, w, R_b^*)$ from $\mathcal{C}$. $\mathcal{B}$ uses its key-derivation oracle to obtain a trapdoor $T_{w'}$ (a private key $d_{w'}$ corresponding to an identity w') for a keyword w'.

Case I: If $\mathbf{Dec}(C^*, T_{w'}) = R_{b'}^*$ (for some $b' = 0, 1$) then $\mathcal{B}$ wins attacking IBPRE-IND-CPA security of **IBPRE**. Hence, if $\mathbf{Dec}(C^*, T_{w'}) = R_1^*$ then it returns 1 else it returns 0. Otherwise, we can assume that there exists $R^* \in \mathcal{M}$ $(R^* \neq R_0^*, R_1^*)$ such that $\mathbf{Dec}(C_2^*, T_{w'}) = R^*$. It is contradict to $\mathbf{Test}(\mathsf{CT}_w, T_{w'}) = 1$ in the definition of the consistency of KU-PEKS scheme.

Case II: Suppose that $\mathbf{Dec}(C^*, T_{w'}) = R_{b'}^*$ (for some $b' = 0, 1$). There exist a keyword $w_0 \in \{0,1\}^*$ and $R^* \in \mathcal{M}$ such that $\mathsf{CT}_{w_0} = \mathbf{PEKS}(\mathsf{PP}, w_0) = [R^*, \mathbf{Enc}(\mathsf{PP}, w_0, R^*)]$, $T_{w_0} = \mathbf{Td}(\mathsf{msk}, w_0)$, $\mathsf{kuTd}_{w_0 \to w} = \mathbf{kuTd}(\mathsf{PP}, T_{w_0}, w_0, w)$, $\mathsf{CT}^* = [R_{b'}^*, C^*] = \mathbf{kuPEKS}\ (\mathsf{PP}, \mathsf{CT}_{w_0}, \mathsf{kuTd}_{w_0 \to w})$, and $T_{w'} = \mathbf{Td}(\mathsf{msk}, w')$.

Suppose that $R^* \neq R_0^*, R_1^*$ and $\mathbf{Reencryt}(\mathsf{PP}, \mathbf{Enc}(\mathsf{PP}, w_0, R^*), \mathsf{kuTd}_{w_0 \to w}) = C^*$. If $\mathbf{Dec}(C^*, T_{w'}) = R_{b'}^*$ (for some $b' = 0, 1$) then it is contradict to the definition of the correctness of IBPRE scheme. Meanwhile, if $\mathbf{Dec}(C^*, T_{w'}) = R_{b'}^*$ (for some $b' = 0, 1$) then $\mathcal{B}$ wins attacking IBPRE-IND-CPA security of **IBPRE**. Hence, if $\mathbf{Dec}(C^*, T_{w'}) = R_1^*$ then it returns 1 else it returns 0. It is easy to see that

$$\Pr[\mathbf{Exp}^{\mathsf{ibpre\text{-}ind\text{-}cpa\text{-}1}}_{\mathbf{IBPRE},\mathcal{B}}(k) = 1] \geq \Pr[\mathbf{Exp}^{\mathsf{kupeks\text{-}cons}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(k) = 1],$$
$$\Pr[\mathbf{Exp}^{\mathsf{ibpre\text{-}ind\text{-}cpa\text{-}0}}_{\mathbf{IBPRE},\mathcal{B}}(k) = 1] \leq 2^{-k}.$$

Therefore, $\mathbf{Adv}^{\mathsf{kupeks\text{-}cons}}_{\mathbf{KU\text{-}PEKS},\mathcal{A}}(k) \leq \mathbf{Adv}^{\mathsf{ibpre\text{-}ind\text{-}cpa}}_{\mathbf{IBPRE},\mathcal{B}}(k) + 2^{-k}$.

5 Applications

As efforts to improve an efficiency of database management, the research of relational database management system (RDBMS) is very mentally active. To convert from one keyword into another keyword in an encrypted email system, we proposed a keyword-updatable PEKS scheme. This had improved the problem of keyword inconsistency residing in PEKS system processed by guessable keywords without previous commitments. After converting the ciphertexts, if only the converted ciphertexts remain open over the public database then these ciphertexts confidentiality and the corresponding queries privacy against an outside adversary also are provided.

References

1. Abdalla, M., Bellare, M., Catalano, D., Kiltz, E., Kohno, T., Lange, T., Malone-Lee, J., Neven, G., Paillier, P., Shi, H.: Searchable encryption revisited: consistency properties. Relat. Anonymous IBE, Extensions, J. Crypt. **21**(3), 350–391 (2008)
2. Ateniese, G., Benson, K., Hohenberger, S.: Key-Private proxy re-encryption. In: Fischlin, M. (ed.) CT-RSA 2009. LNCS, vol. 5473, pp. 279–294. Springer, Heidelberg (2009)
3. Bellare, M., Boldyreva, A., Desai, A., Pointcheval, D.: Key-Privacy in public-key encryption. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 566–582. Springer, Heidelberg (2001)
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
5. Blundo, C., Iovino, V., Persiano, G.: Private-Key hidden vector encryption with key confidentiality. In: Garay, J.A., Miyaji, A., Otsuka, A. (eds.) CANS 2009. LNCS, vol. 5888, pp. 259–277. Springer, Heidelberg (2009)
6. Baek, J., Safavi-Naini, R., Susilo, W.: Public key encryption with keyword search revisited. In: Proceedings of ACIS2006 (2006)
7. Boyen, X., Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles). In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 290–307. Springer, Heidelberg (2006)
8. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
9. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM **45**(6), 965–981 (1998)
10. Green, M., Ateniese, G.: Identity-based proxy re-encryption. In: Katz, J., Yung, M. (eds.) ACNS 2007. LNCS, vol. 4521, pp. 288–306. Springer, Heidelberg (2007)
11. Gentry, C., Silverberg, A.: Hierarchical ID-Based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
12. Garg, S., Sahai, A., Waters, B.: Efficient fully collusion-resilient traitor tracing scheme, Cryptology ePrint Archive, in report /532 (2009). http://eprint.iacr.org/2009/532/
13. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)

14. Jeong, I.R., Kwon, J.O., Hong, D., Lee, D.H.: Constructing PEKS schemes secure against keyword guessing attacks is possible? Elsevier's Comput. Commun. **32**(2), 394–396 (2009)
15. Kiltz, E., Galindo, D.: Direct chosen-ciphertext secure identity-based key encapsulation without random oracle. J. Theor. Comput. Sci. **410**(47–49), 5093–5111 (2009)
16. Koo, W., Hwang, J., Lee, D.: Collusion-resistatn identity-based proxy re-encryption scheme. In: Proceedings of International Conference, Information Science and Technology, pp. 265–269 (2012)
17. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)
18. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Improved searchable public key encryption with designated tester. In: Proceedings of ASIACCS, pp. 376–379 (2009)
19. Rhee, H.S., Park, J.H., Susilo, W., Lee, D.H.: Trapdoor security in a searchable public-key encryption scheme with a designated tester. J. Syst. Softw. **83**(5), 763–771 (2010)
20. Seo, J.H., Kobayashi, T., Ohkubo, M., Suzuki, K.: Anonymous hierarchical identity-based encryption with constant size ciphertexts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 215–234. Springer, Heidelberg (2009)
21. Shen, E., Shi, E., Waters, B.: Predicate privacy in encryption systems. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 457–473. Springer, Heidelberg (2009)