# Fuzzy Keyword Search over Encrypted Data in Cloud Computing

Jin Li[†], Qian Wang[†], Cong Wang[†], Ning Cao[‡], Kui Ren[†], and Wenjing Lou[‡]
[†]Department of ECE, Illinois Institute of Technology
[‡]Department of ECE, Worcester Polytechnic Institute
Email: [†]{jinli, qian, cong, kren}@ece.iit.edu, [‡]{ncao, wjlou}@ece.wpi.edu

*Abstract*—**As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud. For the protection of data privacy, sensitive data usually have to be encrypted before outsourcing, which makes effective data utilization a very challenging task. Although traditional searchable encryption schemes allow a user to securely search over encrypted data through keywords and selectively retrieve files of interest, these techniques support only *exact* keyword search. That is, there is no tolerance of minor typos and format inconsistencies which, on the other hand, are typical user searching behavior and happen very frequently. This significant drawback makes existing techniques unsuitable in Cloud Computing as it greatly affects system usability, rendering user searching experiences very frustrating and system efficacy very low. In this paper, for the first time we formalize and solve the problem of effective fuzzy keyword search over encrypted cloud data while maintaining keyword privacy. Fuzzy keyword search greatly enhances system usability by returning the matching files when users' searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when *exact* match fails. In our solution, we exploit edit distance to quantify keywords similarity and develop an advanced technique on constructing fuzzy keyword sets, which greatly reduces the storage and representation overheads. Through rigorous security analysis, we show that our proposed solution is secure and privacy-preserving, while correctly realizing the goal of fuzzy keyword search.**

## I. Introduction

As Cloud Computing becomes prevalent, more and more sensitive information are being centralized into the cloud, such as emails, personal health records, government documents, etc. By storing their data into the cloud, the data owners can be relieved from the burden of data storage and maintenance so as to enjoy the on-demand high quality data storage service. However, the fact that data owners and cloud server are not in the same trusted domain may put the oursourced data at risk, as the cloud server may no longer be fully trusted. It follows that sensitive data usually should be encrypted prior to outsourcing for data privacy and combating unsolicited accesses. However, data encryption makes effective data utilization a very challenging task given that there could be a large amount of outsourced data files. Moreover, in Cloud Computing, data owners may share their outsourced data with a large number of users. The individual users might want to only retrieve certain specific data files they are interested in during a given session. One of the most popular ways is to selectively retrieve files through keyword-based search instead of retrieving all the encrypted files back which is completely impractical in cloud computing scenarios. Such keyword-based search technique allows users to selectively retrieve files of interest and has been widely applied in plaintext search scenarios, such as Google search [1]. Unfortunately, data encryption restricts user's ability to perform keyword search and thus makes the traditional plaintext search methods unsuitable for Cloud Computing. Besides this, data encryption also demands the protection of keyword privacy since keywords usually contain important information related to the data files. Although encryption of keywords can protect keyword privacy, it further renders the traditional plaintext search techniques useless in this scenario.

To securely search over encrypted data, searchable encryption techniques have been developed in recent years [2]–[10]. Searchable encryption schemes usually build up an index for each keyword of interest and associate the index with the files that contain the keyword. By integrating the trapdoors of keywords within the index information, effective keyword search can be realized while both file content and keyword privacy are well-preserved. Although allowing for performing searches securely and effectively, the existing searchable encryption techniques do not suit for cloud computing scenario since they support only *exact* keyword search. That is, there is no tolerance of minor typos and format inconsistencies. It is quite common that users' searching input might not exactly match those pre-set keywords due to the possible typos, such as `Illinois` and `Ilinois`, representation inconsistencies, such as `PO BOX` and `P.O. Box`, and/or her lack of exact knowledge about the data. The naive way to support fuzzy keyword search is through simple spell check mechanisms. However, this approach does not completely solve the problem and sometimes can be ineffective due to the following reasons: on the one hand, it requires additional interaction of user to determine the correct word from the candidates generated by the spell check algorithm, which unnecessarily costs user's extra computation effort; on the other hand, in case that user accidentally types some other valid keywords by mistake (for example, search for "hat" by carelessly typing "cat"), the spell check algorithm would not even work at all, as it can never differentiate between two actual valid words. Thus, the drawbacks of existing schemes signifies the important need for new techniques that support searching flexibility, tolerating both minor typos and format inconsistencies.

In this paper, we focus on enabling effective yet privacy-preserving fuzzy keyword search in Cloud Computing. To the best of our knowledge, we formalize for the first time the problem of effective fuzzy keyword search over encrypted cloud data while maintaining keyword privacy. Fuzzy keyword search greatly enhances system usability by returning the matching files when users' searching inputs exactly match the predefined keywords or the closest possible matching files based on keyword similarity semantics, when *exact* match fails. More specifically, we use edit distance to quantify keywords similarity and develop a novel technique, i.e., an wildcard-based technique, for the construction of fuzzy keyword sets. This technique eliminates the need for enumerating all the fuzzy keywords and the resulted size of the fuzzy keyword sets is significantly reduced. Based on the constructed fuzzy keyword sets, we propose an efficient fuzzy keyword search scheme. Through rigorous security analysis, we show that the proposed solution is secure and privacy-preserving, while correctly realizing the goal of fuzzy keyword search.

The rest of paper is organized as follows: Section II summarizes the features of related work. Section III introduces the system model, threat model, our design goal and briefly describes some necessary background for the techniques used in this paper. Section IV shows a straightforward construction of fuzzy keyword search scheme. Section V provides the detailed description of our proposed schemes, including the efficient constructions of fuzzy keyword set and fuzzy keyword search scheme. Section VI presents the security analysis. Finally, Section VIII concludes the paper.

## II. Related Work

**Plaintext fuzzy keyword search.** Recently, the importance of fuzzy search has received attention in the context of plaintext searching in information retrieval community [11]–[13]. They addressed this problem in the traditional information-access paradigm by allowing user to search without using try-and-see approach for finding relevant information based on approximate string matching. At the first glance, it seems possible for one to directly apply these string matching algorithms to the context of searchable encryption by computing the trapdoors on a character base within an alphabet. However, this trivial construction suffers from the dictionary and statistics attacks and fails to achieve the search privacy.

**Searchable encryption.** Traditional searchable encryption [2]–[8], [10] has been widely studied in the context of cryptography. Among those works, most are focused on efficiency improvements and security definition formalizations. The first construction of searchable encryption was proposed by Song et al. [3], in which each word in the document is encrypted independently under a special two-layered encryption construction. Goh [4] proposed to use Bloom filters to construct the indexes for the data files. To achieve more efficient search, Chang et al. [7] and Curtmola et al. [8] both proposed similar "index" approaches, where a single encrypted hash table index is built for the entire

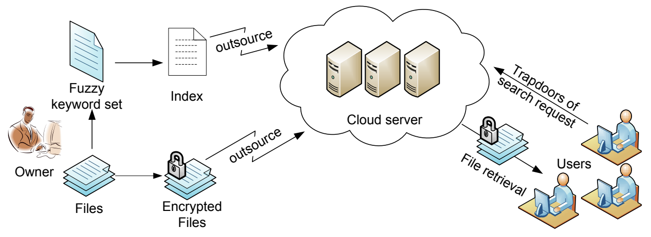

Fig. 1: Architecture of the fuzzy keyword search

file collection. In the index table, each entry consists of the trapdoor of a keyword and an encrypted set of file identifiers whose corresponding data files contain the keyword. As a complementary approach, Boneh et al. [5] presented a public-key based searchable encryption scheme, with an analogous scenario to that of [3]. Note that all these existing schemes support only exact keyword search, and thus are not suitable for Cloud Computing.

**Others.** Private matching [14], as another related notion, has been studied mostly in the context of secure multiparty computation to let different parties compute some function of their own data collaboratively without revealing their data to the others. These functions could be intersection or approximate private matching of two sets, etc. The private information retrieval [15] is an often-used technique to retrieve the matching items secretly, which has been widely applied in information retrieval from database and usually incurs unexpectedly computation complexity.

## III. Problem Formulation

### A. System Model

In this paper, we consider a cloud data system consisting of data owner, data user and cloud server. Given a collection of $n$ encrypted data files $\mathcal{C} = (\mathrm{F}_1, \mathrm{F}_2, \ldots, \mathrm{F}_\mathrm{N})$ stored in the cloud server, a predefined set of distinct keywords $W = \{w_1, w_2, ..., w_p\}$, the cloud server provides the search service for the authorized users over the encrypted data $\mathcal{C}$. We assume the authorization between the data owner and users is appropriately done. An authorized user types in a request to selectively retrieve data files of his/her interest. The cloud server is responsible for mapping the searching request to a set of data files, where each file is indexed by a file ID and linked to a set of keywords. The fuzzy keyword search scheme returns the search results according to the following rules: 1) if the user's searching input exactly matches the pre-set keyword, the server is expected to return the files containing the keyword[1]; 2) if there exist typos and/or format inconsistencies in the searching input, the server will return the closest possible results based on pre-specified similarity semantics (to be formally defined in section III-D). An architecture of fuzzy keyword search is shown in the Fig. 1.

[1]Note that we do not differentiate between files and file IDs in this paper.

## B. Threat Model

We consider a semi-trusted server. Even though data files are encrypted, the cloud server may try to derive other sensitive information from users' search requests while performing keyword-based search over $\mathcal{C}$. Thus, the search should be conducted in a secure manner that allows data files to be securely retrieved while revealing as little information as possible to the cloud server. In this paper, when designing fuzzy keyword search scheme, we will follow the security definition deployed in the traditional searchable encryption [8]. More specifically, it is required that nothing should be leaked from the remotely stored files and index beyond the outcome and the pattern of search queries.

## C. Design Goals

In this paper, we address the problem of supporting efficient yet privacy-preserving fuzzy keyword search services over encrypted cloud data. Specifically, we have the following goals: i) to explore new mechanism for constructing storage-efficient fuzzy keyword sets; ii) to design efficient and effective fuzzy search scheme based on the constructed fuzzy keyword sets; iii) to validate the security of the proposed scheme.

## D. Preliminaries

**Edit Distance** There are several methods to quantitatively measure the string similarity. In this paper, we resort to the well-studied edit distance [16] for our purpose. The edit distance $\mathsf{ed}(w_1, w_2)$ between two words $w_1$ and $w_2$ is the number of operations required to transform one of them into the other. The three primitive operations are 1) Substitution: changing one character to another in a word; 2) Deletion: deleting one character from a word; 3) Insertion: inserting a single character into a word. Given a keyword $w$, we let $S_{w,d}$ denote the set of words $w'$ satisfying $\mathsf{ed}(w, w') \leq d$ for a certain integer $d$.

**Fuzzy Keyword Search** Using edit distance, the definition of fuzzy keyword search can be formulated as follows: Given a collection of $n$ encrypted data files $\mathcal{C} = (F_1, F_2, \ldots, F_N)$ stored in the cloud server, a set of distinct keywords $W = \{w_1, w_2, ..., w_p\}$ with predefined edit distance $d$, and a searching input $(w, k)$ with edit distance $k$ $(k \leq d)$, the execution of fuzzy keyword search returns a set of file IDs whose corresponding data files possibly contain the word $w$, denoted as $FID_w$: if $w = w_i \in W$, then return $FID_{w_i}$; otherwise, if $w \notin W$, then return $\{FID_{w_i}\}$, where $\mathsf{ed}(w, w_i) \leq k$. Note that the above definition is based on the assumption that $k \leq d$. In fact, $d$ can be different for distinct keywords and the system will return $\{FID_{w_i}\}$ satisfying $\mathsf{ed}(w, w_i) \leq \min\{k, d\}$ if exact match fails.

## IV. The Straightforward Approach

Before introducing our construction of fuzzy keyword sets, we first propose a straightforward approach that achieves all the functions of fuzzy keyword search, which aims at providing an overview of how fuzzy search scheme works over encrypted data.

Assume $\Pi$=($\mathsf{Setup}(1^\lambda)$, $\mathsf{Enc}(sk, \cdot)$, $\mathsf{Dec}(sk, \cdot)$) is a symmetric encryption scheme, where $sk$ is a secret key, $\mathsf{Setup}(1^\lambda)$ is the setup algorithm with security parameter $\lambda$, $\mathsf{Enc}(sk, \cdot)$ and $\mathsf{Dec}(sk, \cdot)$ are the encryption and decryption algorithms, respectively. Let $T_{w_i}$ denote a trapdoor of keyword $w_i$. Trapdoors of the keywords can be realized by applying a one-way function $f$, which is similar as [2], [4], [8]: Given a keyword $w_i$ and a secret key $sk$, we can compute the trapdoor of $w_i$ as $T_{w_i} = f(sk, w_i)$.

The scheme of the fuzzy keyword search goes as follows:

We begin by constructing the fuzzy keyword set $S_{w_i,d}$ for each keyword $w_i \in W$ $(1 \leq i \leq p)$ with edit distance $d$. The intuitive way to construct the fuzzy keyword set of $w_i$ is to enumerate all possible words $w'_i$ that satisfy the similarity criteria $\mathsf{ed}(w_i, w'_i) \leq d$, that is, all the words with edit distance $d$ from $w_i$ are listed. For example, the following is the listing variants after a substitution operation on the first character of keyword `CASTLE`: {`AASTLE`, `BASTLE`, `DASTLE`, $\cdots$, `YASTLE`, `ZASTLE`}. Based on the resulted fuzzy keyword sets, the fuzzy search over encrypted data is conducted as follows:

1) To build an index for $w_i$, the data owner computes trapdoors $T_{w'_i} = f(sk, w'_i)$ for each $w'_i \in S_{w_i,d}$ with a secret key $sk$ shared between data owner and authorized users. The data owner also encrypts $\mathrm{FID}_{w_i}$ as $\mathsf{Enc}(sk, \mathrm{FID}_{w_i} \| w_i)$. The index table $\{(\{T_{w'_i}\}_{w'_i \in S_{w_i,d}}, \mathsf{Enc}(sk, \mathrm{FID}_{w_i} \| w_i))\}_{w_i \in W}$ and encrypted data files are outsourced to the cloud server for stroage; 2) To search with $w$, the authorized user computes the trapdoor $T_w$ of $w$ and sends it to the server; 3) Upon receiving the search request $T_w$, the server compares it with the index table and returns all the possible encrypted file identifiers $\{\mathsf{Enc}(sk, \mathrm{FID}_{w_i} \| w_i)\}$ according to the fuzzy keyword definition in section III-D. The user decrypts the returned results and retrieves relevant files of interest.

This straightforward approach apparently provides fuzzy keyword search over the encrypted files while achieving search privacy using the technique of secure trapdoors. However, this approach has serious efficiency disadvantages. The simple enumeration method in constructing fuzzy keyword sets would introduce large storage complexities, which greatly affect the usability. Recall that in the definition of edit distance, substitution, deletion and insertion are three kinds of operations in computation of edit distance. The numbers of all similar words of $w_i$ satisfying $\mathsf{ed}(w_i, w'_i) \leq d$ for $d = 1, 2$ and 3 are approximately $2k \times 26$, $2k^2 \times 26^2$, and $\frac{4}{3}k^3 \times 26^3$, respectively. For example, assume there are $10^4$ keywords in the file collection with average keyword length 10, $d = 2$, and the output length of hash function is 160 bits, then, the resulted storage cost for the index will be 30GB. Therefore, it brings forth the demand for fuzzy keyword sets with smaller size.

## V. Constructions of Effective Fuzzy Keyword Search in Cloud

The key idea behind our secure fuzzy keyword search is two-fold: 1) building up fuzzy keyword sets that incorporate not only the exact keywords but also the ones differing slightly due to minor typos, format inconsistencies, etc.; 2) designing an efficient and secure searching approach for file retrieval based on the resulted fuzzy keyword sets.

### *A. Advanced Technique for Constructing Fuzzy Keyword Sets*

To provide more practical and effective fuzzy keyword search constructions with regard to both storage and search efficiency, we now propose an advanced technique to improve the straightforward approach for constructing the fuzzy keyword set. Without loss of generality, we will focus on the case of edit distance $d = 1$ to elaborate the proposed advanced technique. For larger values of $d$, the reasoning is similar. Note that the technique is carefully designed in such a way that while suppressing the fuzzy keyword set, it will not affect the search correctness.

**Wildcard-based Fuzzy Set Construction** In the above straightforward approach, all the variants of the keywords have to be listed even if an operation is performed at the same position. Based on the above observation, we proposed to use a wildcard to denote edit operations at the same position. The wildcard-based fuzzy set of $w_i$ with edit distance $d$ is denoted as $S_{w_i,d}$=$\{S'_{w_i,0}, S'_{w_i,1}, \cdots, S'_{w_i,d}\}$, where $S'_{w_i,\tau}$ denotes the set of words $w'_i$ with $\tau$ wildcards. Note each wildcard represents an edit operation on $w_i$. For example, for the keyword CASTLE with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as $S_{\texttt{CASTLE},1} = \{$CASTLE, ⋆CASTLE, ⋆ASTLE, C⋆ASTLE, C⋆STLE, $\cdots$, CASTL⋆E, CASTL⋆, CASTLE⋆$\}$. The total number of variants on CASTLE constructed in this way is only $13 + 1$, instead of $13 \times 26 + 1$ as in the above exhaustive enumeration approach when the edit distance is set to be 1. Generally, for a given keyword $w_i$ with length $\ell$, the size of $S_{w_i,1}$ will be only $2\ell + 1 + 1$, as compared to $(2\ell + 1) \times 26 + 1$ obtained in the straightforward approach. The larger the pre-set edit distance, the more storage overhead can be reduced: with the same setting of the example in the straightforward approach, the proposed technique can help reduce the storage of the index from 30GB to approximately 40MB. In case the edit distance is set to be 2 and 3, the size of $S_{w_i,2}$ and $S_{w_i,3}$ will be $C^1_{\ell+1}$+$C^1_\ell \cdot C^1_\ell$+$2C^2_{\ell+2}$ and $C^1_\ell + C^3_\ell + 2C^2_\ell + 2C^2_\ell \cdot C^1_\ell$. In other words, the number is only $O(\ell^d)$ for the keyword with length $\ell$ and edit distance $d$.

### *B. The Efficient Fuzzy Keyword Search Scheme*

Based on the storage-efficient fuzzy keyword sets, we show how to construct an efficient and effective fuzzy keyword search scheme. The scheme of the fuzzy keyword search goes as follows:

1) To build an index for $w_i$ with edit distance $d$, the data owner first constructs a fuzzy keyword set $S_{w_i,d}$ using the wildcard based technique. Then he computes trapdoor set $\{T_{w'_i}\}$ for each $w'_i \in S_{w_i,d}$ with a secret key $sk$ shared between data owner and authorized users. The data owner encrypts $\text{FID}_{w_i}$ as $\mathsf{Enc}(sk, \text{FID}_{w_i} \| w_i)$. The index table $\{(\{T_{w'_i}\}_{w'_i \in S_{w_i,d}}, \mathsf{Enc}(sk, \text{FID}_{w_i} \| w_i))\}_{w_i \in W}$ and encrypted data files are outsourced to the cloud server for storage;

2) To search with $(w, k)$, the authorized user computes the trapdoor set $\{T_{w'}\}_{w' \in S_{w,k}}$, where $S_{w,k}$ is also derived from the wildcard-based fuzzy set construction. He then sends $\{T_{w'}\}_{w' \in S_{w,k}}$ to the server;

3) Upon receiving the search request $\{T_{w'}\}_{w' \in S_{w,k}}$, the server compares them with the index table and returns all the possible encrypted file identifiers $\{\mathsf{Enc}(sk, \text{FID}_{w_i} \| w_i)\}$ according to the fuzzy keyword definition in section III-D. The user decrypts the returned results and retrieves relevant files of interest.

In this construction, the technique of constructing search request for $w$ is the same as the construction of index for a keyword. As a result, the search request is a trapdoor set based on $S_{w,k}$, instead of a single trapdoor as in the straightforward approach. In this way, the searching result correctness can be ensured.

## VI. Security Analysis

In this section, we analyze the correctness and security of the proposed fuzzy keyword search scheme. At first, we show the correctness of the schemes in terms of two aspects, that is, completeness and soundness.

***Theorem** 1:* The wildcard-based scheme satisfies both completeness and soundness. Specifically, upon receiving the request of $w$, all of the keywords $\{w_i\}$ will be returned if and only if $\mathsf{ed}(w, w_i) \le k$.

The proof of this Theorem can be reduced to the following Lemma:

***Lemma** 1:* The intersection of the fuzzy sets $S_{w_i,d}$ and $S_{w,k}$ for $w_i$ and $w$ is not empty if and only if $\mathsf{ed}(w, w_i) \le k$.

*Proof:* First, we show that $S_{w_i,d} \cap S_{w,k}$ is not empty when $\mathsf{ed}(w, w_i) \le k$. To prove this, it is enough to find an element in $S_{w_i,d} \cap S_{w,k}$. Let $w = a_1 a_2 \cdots a_s$ and $w_i = b_1 b_2 \cdots b_t$, where all these $a_i$ and $b_j$ are single characters. After $\mathsf{ed}(w, w_i)$ edit operations, $w$ can be changed to $w_i$ according to the definition of edit distance. Let $w^* = a^*_1 a^*_2 \cdots a^*_m$, where $a^*_i = a_j$ or $a^*_i = *$ if any operation is performed at this position. Since the edit operation is inverted, from $w_i$, the same positions containing wildcard at $w^*$ will be performed. Because $\mathsf{ed}(w, w_i) \le k$, $w^*$ is included in both $S_{w_i,d}$ and $S_{w,k}$, we get the result that $S_{w_i,d} \cap S_{w,k}$ is not empty.

Next, we prove that $S_{w_i,d} \cap S_{w,k}$ is empty if $\mathsf{ed}(w, w_i) > k$. The proof is given by reduction. Assume there exists an $w^*$ belonging to $S_{w_i,d} \cap S_{w,k}$. We will show that $\mathsf{ed}(w, w_i) \le k$, which reaches a contradiction. First, from the assumption that $w^* \in S_{w_i,d} \cap S_{w,k}$, we can get the number of wildcard in $w^*$, which is denoted by $n^*$, is not greater than $k$. Next, we prove that $\mathsf{ed}(w, w_i) \leq n^*$. We will prove the inequality with induction method. First, we prove it holds when $n^* = 1$. There are nine cases should be considered: If $w^*$ is derived from the operation of deletion from both $w_i$ and $w$, then, $\mathsf{ed}(w_i, w) \leq 1$ because the other characters are the same except the character at the same position. If the operation is deletion from $w_i$ and substitution from $w$, we have $\mathsf{ed}(w_i, w) \leq 1$ because they will be the same after at most one substitution from $w_i$. The other cases can be analyzed in a similar way and are omitted. Now, assuming that it holds when $n^* = \gamma$, we need to prove it also holds when $n^* = \gamma + 1$. If $\hat{w}^* = a_1^* a_2^* \cdots a_n^* \in S_{w_i,d} \cap S_{w,k}$, where $a_i^* = a_j$ or $a_i^* = *$. For a wildcard at position $t$, cancel the underlying operations and revert it to the original characters in $w_i$ and $w$ at this position. Assume two new elements $w_i^*$ and $w^*$ are derived from them respectively. Then perform one operation at position $t$ of $w_i^*$ to make the character of $w_i$ at this position be the same with $w$, which is denoted by $w_i'$. After this operation, $w_i^*$ will be changed to $w^*$, which has only $k$ wildcards. Therefore, we have $\mathsf{ed}(w_i', w) \leq \gamma$ from the assumption. We know that $\mathsf{ed}(w_i', w) \leq \gamma$ and $\mathsf{ed}(w_i', w_i) = 1$, based on which we know that $\mathsf{ed}(w_i, w) \leq \gamma + 1$. Thus, we can get $\mathsf{ed}(w, w_i) \leq n^*$. It renders the contradiction $\mathsf{ed}(w, w_i) \leq k$ because $n^* \leq k$. Therefore, $S_{w_i,d} \cap S_{w,k}$ is empty if $\mathsf{ed}(w, w_i) > k$. ■

***Theorem** 2:* The fuzzy keyword search scheme is secure regarding the search privacy.

*Proof:* In the wildcard-based scheme, the computation of index and request of the same keyword is identical. Therefore, we only need to prove the index privacy by using reduction. Suppose the searchable encryption scheme fails to achieve the index privacy against the indistinguishability under the chosen keyword attack, which means there exists an algorithm $\mathcal{A}$ who can get the underlying information of keyword from the index. Then, we build an algorithm $\mathcal{A}'$ that utilizes $\mathcal{A}$ to determine whether some function $f'(\cdot)$ is a pseudo-random function such that $f'(\cdot)$ is equal to $f(sk, \cdot)$ or a random function. $\mathcal{A}'$ has an access to an oracle $\mathcal{O}_{f'(\cdot)}$ that takes as input secret value $x$ and returns $f'(x)$. Upon receiving any request of the index computation, $\mathcal{A}'$ answers it with request to the oracle $\mathcal{O}_{f'(\cdot)}$. After making these trapdoor queries, the adversary outputs two challenge keywords $w_0^*$ and $w_1^*$ with the same length and edit distance, which can be relaxed by adding some redundant trapdoors. $\mathcal{A}'$ picks one random $b \in \{0, 1\}$ and sends $w_b^*$ to the challenger. Then, $\mathcal{A}'$ is given a challenge value $y$, which is either computed from a pseudo-random function $f(sk, \cdot)$ or a random function. $\mathcal{A}'$ sends $y$ back to $\mathcal{A}$, who answers with $b' \in \{0, 1\}$. Suppose $\mathcal{A}$ guesses $b$ correctly with non-negligible probability, which indicates that the value is not randomly computed. Then, $\mathcal{A}'$ makes a decision that $f'(\cdot)$ is a pseudo-random function. As a result, based on the assumption of the indistinguishability of the pseudo-random function from some real random function, $\mathcal{A}$ at most guesses $b$ correctly with approximate probability $1/2$. Thus, the search privacy is obtained. ■

## VII. Conclusion

In this paper, for the first time we formalize and solve the problem of supporting efficient yet privacy-preserving fuzzy search for achieving effective utilization of remotely stored encrypted data in Cloud Computing. We design an advanced technique (i.e., wildcard-based technique) to construct the storage-efficient fuzzy keyword sets by exploiting a significant observation on the similarity metric of edit distance. Based on the constructed fuzzy keyword sets, we further propose an efficient fuzzy keyword search scheme. Through rigorous security analysis, we show that our proposed solution is secure and privacy-preserving, while correctly realizing the goal of fuzzy keyword search.

As our ongoing work, we will continue to research on security mechanisms that support: 1) search semantics that takes into consideration conjunction of keywords, sequence of keywords, and even the complex natural language semantics to produce highly relevant search results; and 2) search ranking that sorts the searching results according to the relevance criteria.

## References

[1] Google, "Britney spears spelling correction," Referenced online at http://www.google.com/jobs/britney.html, June 2009.

[2] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proceedings of Crypto 2007, volume 4622 of LNCS*. Springer-Verlag, 2007.

[3] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. of IEEE Symposium on Security and Privacy'00*, 2000.

[4] E.-J. Goh, "Secure indexes," Cryptology ePrint Archive, Report 2003/216, 2003, http://eprint.iacr.org/.

[5] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. of EUROCRYP'04*, 2004.

[6] B. Waters, D. Balfanz, G. Durfee, and D. Smetters, "Building an encrypted and searchable audit log," in *Proc. of 11th Annual Network and Distributed System*, 2004.

[7] Y.-C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. of ACNS'05*, 2005.

[8] R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: improved definitions and efficient constructions," in *Proc. of ACM CCS'06*, 2006.

[9] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. of TCC'07*, 2007, pp. 535–554.

[10] F. Bao, R. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. of ISPEC'08*, 2008.

[11] C. Li, J. Lu, and Y. Lu, "Efficient merging and filtering algorithms for approximate string searches," in *Proc. of ICDE'08*, 2008.

[12] A. Behm, S. Ji, C. Li, , and J. Lu, "Space-constrained gram-based indexing for efficient approximate string search," in *Proc. of ICDE'09*.

[13] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *Proc. of WWW'09*, 2009.

[14] J. Feigenbaum, Y. Ishai, T. Malkin, K. Nissim, M. Strauss, and R. N. Wright, "Secure multiparty computation of approximations," in *Proc. of ICALP'01*.

[15] R. Ostrovsky, "Software protection and simulations on oblivious rams," Ph.D dissertation, Massachusetts Institute of Technology, 1992.

[16] V. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones," *Problems of Information Transmission*, vol. 1, no. 1, pp. 8–17, 1965.