A Verifiable Fuzzy Keyword Search Scheme Over Encrypted Data

Jianfeng Wang
Department of Mathematics,
Xidian University, China
wjf01@163.com

Xiaofeng Chen*
School of Telecommunications Engineering,
Xidian University, China
xfchen@xidian.edu.cn

Hua Ma
Department of Mathematics,
Xidian University, China
ma_hua@126.com

Qiang Tang
DIES, Faculty of EEMCS
University of Twente, the Netherlands
q.tang@utwente.nl

Jin Li
Department of Computer Science,
Guangzhou University, China
lijin@gzhu.edu.cn

Hui Zhu
School of Telecommunications Engineering,
Xidian University, China
zhuhui@xidian.edu.cn

Abstract

As the Cloud Computing technology becomes mature, the need increases rapidly to store sensitive data securely on the cloud server. Since the cloud is not trusted, the data should be stored in an encrypted form at the server. An inherent problem is how to query the encrypted data efficiently. Recently, some searchable encryption schemes have been proposed in the literatures. Although the existing searchable encryption schemes allow a user to search the encrypted data securely without decrypting it, these solutions cannot support the verifiability of the search results. We argue that a cloud server may be selfish in order to save its computation or download bandwidth. For example, it may execute and return only a fraction of search operations and the results honestly. In this paper, we propose a verifiable fuzzy keyword search scheme based on the symbol-tree which not only supports the fuzzy keyword search over encrypted data, but also enjoys the verifiability of the search outcome.

Keywords: Searchable encryption, fuzzy keyword search, verifiability, cloud computing

1 Introduction

As the Cloud Computing technology becomes mature, storage outsourcing is widely used to reduce operational costs or private backups. By outsourcing their data in the cloud, the data owners can obtain high quality data storage services, while reducing the burden of data storage and maintenance. To securely store the outsourced data on an untrusted cloud server, sensitive data should be encrypted before outsourcing. However, it is intractable for the data owner to search the encryption data in the server efficiently. It is desirable to support the searching functionality on the server side, without decrypting the data and loss of data confidentiality. However, a semi-honest-but-curious public cloud server [4] may be selfish in order to save its computation or download bandwidth. In this scenario, the server may execute only a fraction of search operations honestly and return a fraction of search outcomes honestly. Chai et al. [4] firstly addressed this problem and proposed a verifiable keyword search scheme. However, the solution only supports the exact word search. In 2010, Li et al. [11] proposed a fuzzy keyword search

*Corresponding author: Xidian University, Xi'an 710071, P.R.China, Tel: +86-29-88204749 scheme over encrypted data in cloud computing. In this paper, we incorporate the two ideas to achieve a verifiable fuzzy keyword search scheme. Compared with the previous schemes, our solution is more efficient for the real applications. Besides, we proved the proposed scheme can achieve the desired security properties.

1.1 Related Work

Plaintext Fuzzy Keyword Search Recently, plaintext fuzzy keyword search solutions have been proposed [1],[9],[10]. These solutions are based on approximate string matching techniques. 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 Plenty of searchable encryption schemes have been proposed in recent years, which can be categorized into public-key searchable encryption and symmetric searchable encryption. To construct efficient schemes we focus on symmetric searchable encryption (SSE), where the same client stores and retrieves encrypted documents. Traditional searchable encryption has been discussed in [2],[3],[6],[7],[8],[12]. The first practical scheme for searching in encrypted data was proposed by Song et al. [12], in which each word in the document is encrypted independently under a special two-layered encryption construction. Boneh et al. [3] proposed a public key based searchable encryption scheme where a user A can send a key to a server to allow the server to search data items that are encrypted using A's public key for the presence of certain keywords. Goh [8] proposed to use Bloom filters to construct the indexes for the data files. To achieve more efficient search, Chang et al. [5] and Curtmola et al. [7] both proposed similar "index" approaches, where a single encrypted hash table index is built for the entire file collection.

Li et al. [11] proposed a fuzzy keyword search over encrypted data in cloud computing, which utilized the multi-way tree to enhance the search efficiency. However, note that the semi-honest-but-curious cloud server may be selfish in order to save its computation or download bandwidth. It may execute only a fraction of search operations honestly and return a fraction of search outcome honestly. To solve this problem, Chai et al. [4] proposed a verifiable SSE (VSSE) scheme, which ensures that the user can verify the correctness and completeness of the search results.

1.2 Organization

The organization of this paper is as follows. Some preliminaries are given in Section 2. The proposed verifiable fuzzy keyword search scheme and its security analysis are given in Section 3. Finally, conclusions will be made in Section 4.

2 Preliminary

2.1 Notions

$D = (F_1, F_2, \ldots, F_n)$: a set of n encryption documents;
$W = \{\omega_1, \omega_2, \ldots, \omega_p\}$: the set of distinct keywords of D;
ω_i: i-th keyword of W;
$ID\{F_i\}$: the identifier of document F_i;

ID_{ω_i}: the identifiers of document containing the keyword ω_i;
$Enc(sk,\cdot)$ and $Dec(sk,\cdot)$: the encryption and decryption algorithms;
g_k: a keyed hash function;
s_k: a symmetric cipher;
$\{T_{\omega'}\}$: the trapdoor set of all fuzzy keywords of $\omega' \in S_{\omega,d}$;
$\triangle = \{\alpha_i\}$: the predefined symbol set, where $|\triangle| = 2^n$, and $\alpha_i \in \triangle$ can be denoted by n bits;
G_W: a tree covering all the fuzzy keyword of $\omega \in W$ is built up based on symbols in $|\triangle|$;
ε: an sequence alphabetic set of size $|\varepsilon|$;
$T_{x,y}$: the value of the y-th node from left to right of depth of x in G_W;
$ord(T_{x,y}[r_0])$: the alphabetic order of the character $T_{x,y}[r_0]$ in ε;
$parent(T_{j,q_j})$: the parent node of the T_{j,q_j} node.

2.2 Definitions

The verifiable fuzzy keyword search scheme (VFKS) consists of the algorithms (**Keygen**, **Buildindex**, **trapdoor**, **search**), which are similar to those of standard Searchable Symmetric Searchable Encryption scheme (SSE), as well as a new algorithms **Verify**. The algorithms are defined as follows.

- **Keygen**(1^k): This algorithm is run by the user to setup the scheme. It takes a security parameter k as input, and outputs the secret key sk and the document encryption key sk', where the sk is used to generate the index and the sk' is used to encrypt the document.

- **Buildindex**(sk,W): This algorithm is run by the user to create the index. It takes a secret sk and the distinct keyword set of the document collection D as inputs, and outputs a symbol-based tree G_W.

- **Trapdoor**$(sk,S_{\omega,d})$: This algorithm is run by the user to generate trapdoors for all fuzzy keywords of the user input keyword ω. It takes a secret key sk and a fuzzy keyword set $S_{\omega,d}$ as inputs, and outputs a trapdoor set $\{T_{\omega'}\}_{\omega' \in S_{\omega,d}}$.

- **Search**$(G_W,\{T_{\omega'}\})$: This algorithm is run by the server in order to search for the documents in D that contain keyword ω. It takes the symbol-based tree G_W of a document collection D and a trapdoor set $\{T_{\omega'}\}$ of fuzzy keyword set $S_{\omega,d}$ as inputs, and if search is successful outputs $True$, ID_ω and the $proof$, otherwise outputs $False$ and the $proof$.

- **Verify**$(T_\omega, proof)$: This algorithm is run by the user to test whether the server is honest. it takes the result of the algorithm *Seach* as input,(if the process *Search* is successful, it takes ID_ω, T_ω and the corresponding $proof$, otherwise takes T_ω and the corresponding $proof$ as input) and outputs $True$ if the server honestly search, otherwise outputs $False$.

Edit Distance Given two keywords ω_1,ω_2, the edit distance (denoted as $ed(\omega_1,\omega_2)$) between ω_1 and ω_2 is the minimum distance of operations needed to transform ω_1 to ω_2. 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 ω, we let $S_{\omega,d}$ denote the set of words ω' satisfying $ed(\omega,\omega') < d$ for a certain integer d.

Trapdoor of Keywords Trapdoors of the keywords are realized by applying a hash function f as follows: Given a keyword w, we compute the trapdoor of w as $T_\omega = f(sk, \omega)$, where the sk is the user's index generation key.

Wildcard-based Fuzzy Set Construction In the traditional method, all the variants of the keywords have to be listed even if an operation is performed at the same position. By using this way, the resulted storage cost for the index will be very large. To avoid the drawback, we exploit the method of [11]. The method use a wildcard to denote edit operations at the same position. The wildcard-based fuzzy set of ω_i with edit distance d is denoted as $S_{\omega_i,d} = \{S'_{\omega_i,0}, S'_{\omega_i,1}, \cdots, S'_{\omega_i,d}\}$, where $S'_{\omega_i,d}$ denotes the set of words ω'_i with d wildcards. For example, for the keyword *cat* with the pre-set edit distance 1, its wildcard-based fuzzy keyword set can be constructed as $S_{cat,1} = \{cat, \star cat, \star at, c\star at, c\star t, ca\star t, ca\star, cat\star\}$.

2.3 System Model

In this paper, we consider a single-user searchable encryption scheme setting in which the user wishes to store an document collection on an semi-honest-but-curious server, while preserving the ability to search through them. Given a collection of encrypted documents $D = (F_1, F_2, \ldots, F_n)$ and a set of keywords $W = \{\omega_1, \omega_2, \ldots, \omega_p\}$. In the initialization phase, the user generates the symbol-based index tree G_W together with the encrypted documents outsource to the cloud server. Upon receiving the search request by the user, the server maps the searching request to a set of document, where each document is indexed by a document identifier and linked to a set of keywords. On searching, the server returns the search results and a $proof$ to the user. The user can verify the correctness and completeness of search results by the $proof$. The fuzzy keyword search 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 identifier of document containing the keyword; 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.

2.4 Security Model

To facilitate our formal discussions, we make the following assumptions. The user honestly generates his secret key sk and stores securely without being eavesdropped on by an outsider. The cloud server is "semi-honest-but-curious", which is not fully trusted by the the user in the following sense.

(1) The server will not delete the encrypted data documents or index from its storage. We consider the server will honestly follow the protocol specifications in performing the storage operation only.

(2) The server is considered curiously, it may tries to analyze data in its storage and message flows in order to learn additional information.

(3) The server may forge the search results as it may execute only a fraction of search operations honestly. For example, to save its computation or download bandwidth, the server may be: 1) execute only a fraction of the request trapdoors and return all the search results honestly ; 2) execute all of the search operations and return a fraction of the search results honestly, and so on.

2.5 Design Goals

To support verifiable fuzzy keyword search over encrypted cloud data using the above system and threat models, our system design should achieve the following security and performance guarantees: 1) our method should support fuzzy keyword search and enables users to verify the correctness and completeness of search results; 2) our solution should prevent the server from learning additional information. Using the scheme, the user can verifies the correctness and completeness of the search outcomes.

3 A New Verifiable Fuzzy Keyword Search Scheme

In this section, we present the proposed scheme in details and the security analysis. Our scheme consists of five algorithms $(\mathsf{KeyGen}, \mathsf{Buildindex}, \mathsf{trapdoor}, \mathsf{search}, \mathsf{verify})$.

3.1 Construction of the VFKS Scheme

- Keygen

  In this process, the user generate the index generation key and document encryption key. The Keygen is a randomized key generation algorithm, which generate the keys in this way : $sk, sk' \xleftarrow{R} \{0,1\}^k$.

- Buildindex

  In this process, we incorporate the generation method of the symbol-based index in [11] and the creation way of searching tree in [4] to construct a new symbol-based tree G_W. The key idea behind this construction is that all trapdoors sharing a common prefix may have common nodes. The root is associated with an empty set and the symbols in a trapdoor can be recovered in a search from the root to the leaf that ends the trapdoor [11]. All fuzzy keywords in the trie can be found by a depth-first search. Assume $\Delta = \{\alpha_i\}$ is a predefined symbol set,where the number of different symbols is $|\Delta| = 2^n$, that is each symbol $\alpha_i \in \Delta$ can be denoted by n bits. The algorithm works as follows:

  (1) **Initialization**
  - The user scan the D and build W, the set of distinct keywords of D.
  - The user outsouce the encryption document collection D to the server and receive the identifiers of each document(denote as $ID\{F_i\}$). For all document of containing the keyword ω_i, denote the identifier set as $ID_{\omega_i} = ID\{F_1\}\|ID\{F_2\}\ldots,\|ID\{F_i\}$.

  (2) **Build fuzzyset trapdoor**
  - For each keyword $\omega_i \in W$, construct the fuzzy keyword set $S_{\omega_i,d}$ with the wildcard-based method [11].
  - build trapdoor $T_{\omega_i'} = f(sk, \omega_i')$ for each $\omega_i' \in S_{\omega_i,d}$ with the index generation key sk and output trapdoor set $\{T_{\omega_i'}\}$.

  (3) **Build symbol-based index tree**
  - Create a depth of l/n full 2^n-binary tree G_W, where each node contains two attributes $(r_0, r_1) = (null, null)$ in detail and l is the out length of hash function $f(x)$.
  - For each fuzzy keyword $\omega_i' \in S_{\omega_i,d}$, divided $T_{\omega_i'}$ into l/n parts, each n-bits hash value represents a symbol in $\triangle$. Putting all the sequence of symbol filling into the G_w and appending the corresponding identifier $ID_{\omega_i}\|g_k(ID_{\omega_i})$ to the leaf node. The algorithm described in Fig. 1 in detail.

- Trapdoor

  - Input a keyword ω, generate the fuzzy keyword set $S_{\omega,d}$ with the wildcard-based way;

```
Input:
    (1) secret $sk$
    (2) trapdoor set $\{T_{\omega_i'}\}$
Output:
    a symbol-tree $G_W$
    1: Create $G_W$ to be a depth of $|l/n|$ full $|2^n|$-ary tree
    2: $(r_0, r_1) \Leftarrow (null, null)$, for each node
    3: $T_{0,0}[r_0] \Leftarrow root; T_{0,0}[r_1] \Leftarrow Enc(sk, \phi); q_0 \Leftarrow 0$
    4: For each trapdoor $T_{\omega_i'}$ in $\{T_{\omega_i'}\}_{1 \leq i \leq p}$ do
    5:     $T_{\omega_i'} = (T_{\omega_i'}[1], T_{\omega_i'}[2], \ldots, T_{\omega_i'}[l/n])$
    6:     $j = 1$
    7:     while $j < l/n$ do
    8:         Find $q_j \in [q_{j-1} \times |2^n| + 1, (1 + q_{j-1} \times |2^n|)]$ such that $T_{j,q_j}[r_0] = T_{\omega_i'}[j]$;
    9:         If cannot, find $q_j$ such that $T_{j,q_j}$ is empty
   10:         $T_{j,q_j}[r_0] \Leftarrow T_{\omega_i'}[j]$
   11:         $T_{j,q_j}[r_1] = Enc(T_{j,q_j}[r_0], parent(T_{j,q_j})[r_1])$
   12:     End while
   13:     if $j = l/n$
   14:         Find $q_j \in [q_{j-1} \times |2^n| + 1, (1 + q_{j-1} \times |2^n|)]$ such that $T_{j,q_j}[r_0] = T_{\omega_i'}[j]$;
   15:         If cannot, find $q_j$ such that $T_{j,q_j}$ is empty
   16:         $T_{j,q_j}[r_0] \Leftarrow T_{\omega_i'}[j]$
   17:         $T_{j,q_j}[r_1] = Enc(T_{j,q_j}[r_0], parent(T_{j,q_j})[r_1]) \parallel ID_{\omega_i'} \parallel g_k(ID_{\omega_i'})$
   18:     end if
   19: Delete all empty nodes
   20: return $G_W$
```

Figure 1: The build symbol-based tree algorithm

- Compute $T_{\omega'} = f(sk, \omega')$ for each $\omega' \in S_{\omega,d}$ and output trapdoor set $\{T_{\omega'}\}$, meanwhile, the user need to storage the $\{T_{\omega'}\}$ and the number of those, which are used during the verify process.

- Search
  - Upon receiveing the search request, the server divides each $T_{\omega'}$ into a sequence of symbols;
  - Performs the search over G_W using algorithm described in Fig 2 and returns the ID_{ω_i} and *proof* to the user.
  - According to the identifier, the user can get the interest documents.

- Verify
  - check whether the number of received *proof* is equals to the number of sent trapdoor.
  - input the T_ω and the corresponding *proof*, according to the algorithm describe in Fig 3, test whether the server is honest.

```
Input:
    (1) Symbol-Tree $G_W$
    (2) trapdoor set $\{T'_\omega\}$
Output:
    (1) “Yes”,if the search is successful; “No”, otherwise
    (2) document identifiers if “Yes”
    (3) proof of the search outcome
    1: proof $\Leftarrow T_{0,0}[r_1]$; $q_0 \Leftarrow 0$
    2: For $i$ from 1 to $|T'_\omega|$ do
    3: $T\omega_i = \{T\omega_i[1],\dots,T\omega_i[l/n]\}$
    4:   For $j$ from 1 to $|l/n|$ do
    3:      $mark = False$
    4:      For $q_j \in [q_{j-1} \times |2^n|+1,(1+q_{j-1}) \times |2^n|]$ do
    5:          If $T_{j,q_j}[r_0] = T\omega_i[j]$ then
    6:              mark $\Leftarrow$ True; proof $\Leftarrow$ proof $\|$ $(T_{j,q_j}[r_1])$; break
    7:          End If
    8:      End For
    9:      If mark = False then
   10:          proof $\Leftarrow proof \parallel j$
   11:          return “No” and proof
   12:      End If
   13: End For
   14:      proof $\Leftarrow proof \parallel j$
   15: If $T_{j,q_j}$ has no child then
   16:      return “$Yes$”, the last parts of $T_{j,q_j}[r_1]$ as document identifiers and proof
   17: End If
```

Figure 2: The search algorithm of the verifiable fuzzy keyword search

3.2 Security analysis

Data Privacy. In this work, we consider only search privacy, because of privacy of the documents can be ensured by the encryption algorithm. That is, we focus on the confidentiality of the search request and the index T. Using the trapdoor technology, the attacker directly to get the plaintext is impossible from the ciphertext. So we mostly concern the confidentiality of the index T. In [7], Li et al. has proved that the fuzzy keyword search scheme is secure regarding the search privacy. Similar arguments will work for our scheme.

Verifiable Searchability. As [8], we assume k steps are performed by the server. If “No” returned, we would know that the first $k-1$ characters are matched while $T_w[k]$ is mismatched, which could be described by a k bit binary sequence $b = (1,\dots,1,0)$; if “Yes” is returned, $b = (1,\dots,1,1)$.

In our scheme, firstly, we check the number of proof whether is equal to the sending trapdoors. If not, we can say the server is not make a full search. If pass, we exploit a random sampling method to check. For each of Proof to be tested, As [8], Similarly, starting from the last (or k-th) step, if “Yes”, verify checks the integrity of the concatenation of the document identifiers by computing a keyed hash of it and comparing with the received one. In fact, the completeness of the search outcome is examined

```
Input:
    (1) "Yes" with document identifiers or "No"
    (2) proof:$T_{1,q_1}[r_1]||T_{2,q_2}[r_1]||\dots||T_{j,q_j}[r_1]||j$
    (3) trapdoor: $T_\omega = (T_\omega[1],\dots,T_\omega[l/n])$
Output:
    True Or False
    1: if "Yes" b $\Leftarrow 1,\dots,1,1$; otherwise b $\Leftarrow 1,\dots,1,0$
    2: if "Yes" then
    3:     return False if $g_k(I\hat{D}_{\omega_i}) \neq g_k(ID_{\omega_i})$,where $I\hat{D}_{\omega_i}$ is the received identifiers.
    4: end If
    5: if $j = l/n$
    6: decrypt the first part of $T_{j,q_j}[r_1]$ to get (x,y)
    7: if $x \neq T_\omega[j]$ or $y \neq T_\omega[j-1]$
    8:         return False
    9: while $j \geq 0$ do
   10:     $j--$
   11:     decrypt the first part of $T_{j,q_j}[r_1]$ to get (x,y)
   12:     if $x \neq T_\omega[j]$ or $y \neq T_\omega[j-1]$
   13:         return False
   14:     end If
   15: end while
   16: return True
```

Figure 3: The verify algorithm of the verifiable fuzzy keyword search

here. If the server return a fraction of the search outcome, the user can find the server is not honest. Then, we test that whether the trapdoor equals the received symbol string of the proof. After that, j decreased by one. If "No", the above step is skipped. Next, verify validates the correctness of the search outcome by decrypting the first part of $T_{j,q_j}[r_1] = Enc(T_{j,q_j}[r_0], parent(T_{j,q_j})[r_1])$ to get (x,y) and testing whether: (1) $T_\omega[j]$ equals x (2) $T_\omega[j-1]$ equals y. To cheat the search results, the server need to forge the proof. There are two possible case: (1) the server honestly search for a fraction of trapdoors and forge the proof for other trapdoors, at worst, the server do not any search; (2) the server forge the proof according to the received trapdoor. For the case (1), the server must can generate a valid r_1 with a different $m\hat{e}m \neq mem$, consider the adversary do not know the sk, it can be seen a random oracle. In case (2), the adversary may use the r_1 of another node, note that each node has a global unique r_1, which will be rejected by verify. In addition, the argument above can be applied recursively to the $(j-1)-$th step in verify and so on.

4 Conclusion

In this paper, we investigate the fuzzy keyword search problem in the scenario of a semi-honest-but-curious server, which which may execute only a fraction of search operations honestly and return a fraction of search outcome honestly. We first propose a verifiable fuzzy keyword search scheme, which not only supports fuzzy keyword search over encrypted data, but also enjoys the verifiability of the search outcome. In the further work, we continue to research on verifiable fuzzy keyword search scheme that support multi-keyword query.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 60970144 and 61100224) and the Fundamental Research Funds for the Central Universities (Nos. K50510010003, JY10000901034 and K50510010030).

References

[1] A. Behm, S. Ji, C. Li, and J. Lu. Space-constrained gram-based indexing for efficient approximate string search. In *Proc. of the 25th IEEE International Conference on Data Engineering(ICDE'09), Shanghai, China*, pages 604–615. IEEE, March-April 2009.

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques(Eurocrypt'04), Interlaken, Switzerland, LNCS*, volume 3027, pages 506–522. Springer-Verlag, May 2004.

[3] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Proc. of the 4th Theory of Cryptography Conference(TCC'04), Amsterdam, The Netherlands, LNCS*, volume 4392, pages 535–554. Springer-Verlag, February 2007.

[4] Q. Chai and G. Gong. Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. `http://www.cacr.math.uwaterloo.ca/techreports/2011/cacr2011-22.pdf`, 2009.

[5] Y. C. Chang and M. Mitzenmacher. Privacy preserving keyword searches on remote encypted data. In *Proc. of the 3rd Applied Cryptography and Network Security(ACNS'05), Columbia University, New York, USA, LNCS*, volume 3531, pages 391–421. Springer-Verlag, June 2005.

[6] M. Chuah and W. Hu. Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data. In *Proc. of the 31st International Conference on Distributed Computing Systems Workshops (ICDCSW'11), Minneapolis, Minnesota, USA*, pages 273–281. IEEE, June 2011.

[7] R. Cutmola, J. A. Garay, S. Kamara, and R. Ostrovsky. Searchable symmetric encryption: improved definition and efficient constrcutions. In *Proc. of the 13th ACM Conference on Computer and Communications Security(CCS'06), Alexandria, Virginia, USA*, pages 79–88. ACM press, October-November 2006.

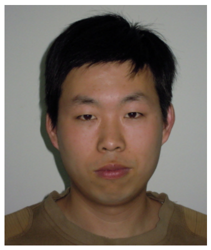
[8] E. J. Goh. Secure indexes. Report 2003/216, Cryptology ePrint Archive, October 2003. `http://eprint.iacr.org/2003/216`.

[9] S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *Proc. of 18th International World Wide Web Conference(WWW'09), Madrid, Spain*. ACM, April 2009.

[10] C. Li, J. Lu, and Y. Lu. Efficient merging and filtering algorithms for approximate string searches. In *Proc. of the 24th IEEE International Conference on Data Engineering(ICDE'08), Cancun, Mexico*, pages 257–266. IEEE, April 2008.

[11] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou. Fuzzy keyword search over encrypted data in cloud computing. In *Proc. of the 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'10), San Diego, CA, USA*, pages 1–5. IEEE, March 2010.

[12] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proc.of the 2000 IEEE Symposium on Security and Privacy(SP'00), Berkeley, California, USA*, pages 44–55. IEEE, May 2000.

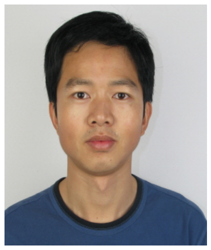


Jianfeng Wang is a graduate student at the Faculty of Science, in Xidian University. His research interests include public key cryptography, cloud computing security and searchable encryption.

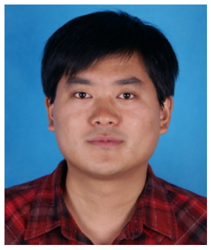
Xiaofeng Chen received his Ph.D. in cryptography from the Xidian University in 2003. He is currently a Professor at the School of Telecommunications Enginnering, Xidian University. His research interests include public key cryptography, financial cryptography, and cloud computing security.

Hua Ma receivered her Master's degree in Applied Mathematics from the Xidian University in 1990. She is currently a Professor at the Faculty of Science, Xidian University. Her research interests include public key cryptography, network Security, and electronic commerce.

Qiang Tang is a postdoc researcher at the Distributed and Embedded Security Research Group in the Computer Science department at University of Twente, the Netherlands. He is leading the Kindred Spirits project and working on the security and privacy issues for online social networks and recommendation systems. Before this, he was a postdoc researcher in the Crypto group at Ecole Normale Superieure, Paris, France. He received his MSc degree from Peking University , China in 2002 and obtained his PhD degree from Royal Holloway, University of London , UK in 2007.



Jin Li was born on Jan. 1981. He got his Ph.D degree from Sun Yat-sen University at 2007. His research interests include design of secure protocols in Cloud Computing and cryptographic protocols. He served as a senior research associate at Korea Advanced Institute of Technology (Korea) and Illinois Institute of Technology (U.S.A.) from 2008 to 2010, respectively. He has published more than 40 papers in international conferences and journals, including IEEE INFOCOM, IEEE Transaction on Parallel and Distributed Computation etc. Currently, he is a professor at Guangzhou University.



Hui Zhu was born in 1981 and received the Ph.D. degree in information security from Xidian University in 2009. Currently, he works as an associate professor at Xidian University. His current research interests include network security and security authentication protocol.