A New Efficient Verifiable Fuzzy Keyword Search Scheme

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

Hua Ma
Department of Mathematics
Xidian University, China
ma$_h$*ua*@126.*com*

Qiang Tang
APSIA group, SnT, University of Luxembourg
6, rue Richard Coudenhove-Kalergi
L-1359 Luxembourg
qiang.tang@uni.lu

Jin Li
School of Computer Science
Guangzhou University, China
jinli71@gmail.com

Hui Zhu
State Key Laboratory of ISN
Network and Data Security Key Laboratory of Sichuan Province
Xidian University, China
zhuhui@xidian.edu.cn

Siqi Ma
School of Computer Science and Technology
Xidian University, China
xdmasiqi@hotmail.com

Xiaofeng Chen*
State Key Laboratory of ISN
Xidian University, China
xfchen@xidian.edu.cn

Abstract

Data outsourcing has attracted considerable attentions recently. It can not only bring high quality data service to users, it also reduces their burden of data storage and maintenance. As the confidentiality of sensitive data, it should be encrypted before outsourcing. To achieve search over the encrypted data becomes a challenging problem. Although the existing searchable encryption schemes allow users to search the encrypted data with confidentiality, these solutions cannot support the verifiability of searching result. We argue that a cloud server may be selfish in order to save its computation ability or bandwidth. For example, it may execute only a fraction of the search and returns part of the searching result. In this paper, we propose a novel verifiable fuzzy keyword search scheme based on the bloom filter which can support the fuzzy keyword search, while enjoying the verifiability of the searching result. Through rigorous security analysis, we show that our proposed scheme is secure under the proposed model, while correctly and efficiently realizing the verifiable fuzzy keyword search.

Keywords: Cloud computing, searchable encryption, verifiable fuzzy keyword search, data privacy

1 Introduction

Cloud computing achieves the long dreamed vision of computing as an infrastructure. It brings the IT services (including computing, storage and bandwidth) from cloud providers to individual users and vendors via Internet, which represents the most rapidly developing area of IT technology and deployment.

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, volume: 3, number: 4, pp. 61-71
*Corresponding author: Xidian University, Xi'an 710071, P.R.China, Tel: +86-29-88204749

As a promise way of data storage in cloud computing environment, data outsourcing has attracted considerable attentions recently. By outsourcing their own data in the cloud, data owners can obtain high quality data storage services, while reducing the burden of local data storage and maintenance. To securely store outsourced data on an untrusted cloud server, sensitive data should be encrypted before outsourcing [1], [2]. In this way, the confidentiality of the outsourced data can achieve even against the insider attackers such as database system administrators. However, it is intractable for data owners to search the encrypted data in the server efficiently. The trivial method of downloading the whole database and decrypting in local is evidently impractical because of the large sum of communication and computation cost [3]. It is desirable to support the searching functionality on the cloud server side, without decrypting the data and loss of data confidentiality. A popular solution is to selectively retrieve files through keyword search instead of retrieving the whole encrypted files back, which is called as searchable encryption.

Since Song proposed the first searchable encryption scheme in [4], there are many researches have been done in recent years. Nevertheless, all of the existing schemes are based on the symmetric searchable encryption scheme [5]. Specially, the cloud is considered as honest-but-curious in these works. That means that the server executes all searching operations and returns all search results honestly, but tries to learn the underlying information of the encrypted data. Note that the cloud may be selfish in order to save its computation ability or bandwidth, we consider a stronger adversary called semi-honest-but-curious server [6]. In this scenario, for his own benefit, the cloud may execute only a fraction of the search and returns part of the searching result honestly for the user. To solve this drawback, Chai et al. [6] firstly proposed a verifiable keyword search scheme using the technology of hash chain. However, the solution only supports the exact keyword search. To achieve more high system usability, Li et al. [7] proposed a searchable encryption scheme support fuzzy keyword search, whereas they have not considered the issue of verifiable keyword search.

1.1 Contributions

In this paper, we propose a new efficient verifiable fuzzy keyword search scheme over the encrypted data in the cloud computing. In the proposed construction, user can perform fuzzy keyword search as well as achieving the searching result. Specifically, our contribution can be summarized as follows:

- We propose a new verifiable fuzzy keyword search (VFKS) scheme by using Using the method of wildcard-based fuzzy keyword set and the Bloom filter, which not only enables fuzzy keyword search over encrypted data, but also maintains keyword privacy and the verifiability of the searching result.

- Through rigorous security analysis, our solution is secure and privacy preserving, while supporting the efficient verifiability of the searching result.

1.2 Related Work

Plaintext Fuzzy Keyword Search Recently, plaintext fuzzy keyword search solutions have been proposed [8],[9],[10]. These solutions are based on approximate string matching techniques, which allow user to search without using try-and-see approach for finding relevant information. At a 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 scheme we focus on symmetric searchable encryption (SSE). Traditional searchable encryption has been discussed in [11],[12],[13],[5],[14],[4]. The first practical scheme for searching in encrypted data was proposed by Song et al. [4], in which each word in the document is encrypted independently under a special two-layered encryption construction. Goh [14] proposed to use Bloom filters to construct the indexes for the data files. To achieve more efficient search, Chang et al. [15] and Curtmola et al. [5] both proposed similar "index" approaches, where a single encrypted hash table index is built for the entire file collection.

As a supplementary method, In 2004, Boneh et al. [12] proposed a public key based searchable encryption scheme(PEKS) 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. Waters et al. [**?**] presented that the PEKS scheme based on the bilinear pairing can be applied to build encrypted and searchable audit logs. To improve the efficiency of PEKS scheme, Baek et al. [**?**] proposed a secure channel-free public key encryption with keyword search scheme without a secure channel between the receiver and the server.

Li et al. [7] 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 ability or bandwidth. It may execute only a fraction of the search and returns part of the searching result honestly. To solve this problem, Chai et al. [6] proposed a verifiable SSE (VSSE) scheme, which ensures that the user can verify the correctness and completeness of the search result.

1.3 Organization

The rest of paper is organized as follows: Section 2 gives some preliminaries. Section 3 provides the detailed description of our proposed scheme. Section 4 presents the security analysis. Finally, Section 5 concludes the paper.

2 Problem Formulation

2.1 Notions

Table 1: Notations

$C = (F_1, F_2, \ldots, F_n)$:	a set of n encryption files
$W = \{w_1, w_2, \ldots, w_p\}$:	the set of distinct keywords of C
ID_{w_i}:	the identifiers of documents containing the keyword w_i
$\triangle = \{\alpha_i\}$:	the predefined symbol set, where $\lvert\triangle\rvert = 2^n$, and $\alpha_i \in \triangle$ can be denoted by n bits
H_i: $\{0,1\}^* \to \{1, \ldots, m\}$:	a Hash function of bloom filter
tk:	trapdoor generation key

2.2 Definitions

Edit Distance Edit distance is a measure of similarity between two strings. The edit distance $ed(w_1, w_2)$ between two words w_1 and w_2 is the minimum number of operations required to transform one to the

other. There are three primitive operations. (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 keywords $w^{'}$ satisfying $ed(w,w^{'}) < d$ for a certain integer d.

Trapdoors 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 $T_w = f(tk,w)$, where the tk is the user's trapdoor generation key.

Bloom Filter Bloom filter is a space-efficient probabilistic data structures that allow for efficient testing of whether or not an item belongs to a set [**?**]. A Bloom filter is essentially a bit array of m bits. Initially, all of When adding an item, one or several hash functions are used to identify bit(s) of the Bloom filter that must be set to 1. To test whether an item belongs to the set, it is hashed again and, if all corresponding bits are set, it is likely to be part of the set; otherwise, it is guaranteed to not belong to the set.

Verifiable of Keyword Search Using pseudo-random function, the server executes search for the user when receiving the search request, and returns the search result and the $proof$. If the server executes all operations honestly, the probability that the search result is incorrect should be negligible; but if the server just returns a fraction of the search result honestly, the user can detect the cheating behavior with overwhelming probability through the $verify$ algorithm.

2.3 System Model

In this paper, we consider a cloud data-outsourcing system, which consists of three different entities: the data owner, the user and the cloud server. The data owner has a collection of n encrypted data files $C = (F_1, F_2, \cdots, F_n)$ to be stored in the cloud server. A predefined set of distinct keywords in C is denoted as $W = (w_1, w_2, \cdots, w_p)$. The cloud server performs fuzzy search for the authorized users over the encrypted data. We assume the authorization between the data owner and users is appropriately done. In the initialization phase, the data owner shares the trapdoor generation key sk with authorized users, and builds an index G_W for the encrypted file collection C together with the encrypted files outsourcing to the cloud server. To search the file collection for any keyword w, an authorized user generates the trapdoor of desirable keyword w, and sends it to the cloud server. Upon receiving the search request by the user, the server performs the search over the index G_W and returns all the encrypted files containing the specific keyword w. For the fuzzy keyword search, the server returns the closest possible results based on pre-specified similarity metrics. An architecture of verifiable fuzzy keyword search is shown in Fig.1.

2.4 Adversary Model

In this work, we consider a "semi-honest-but-curious" cloud server, which is different with most previous searchable encryption schemes. We assume the cloud server acts in an "semi-honest" fashion, that is to say, it may not correctly follow our proposed protocol but forge part of search result or execute only a fraction of searching operations honestly. In addition, the cloud server tries to analyze the message flow received during the protocol in order to learn additional information. When designing verifiable fuzzy keyword search scheme, we follow the security definition deployed in the traditional searchable encryption[5]. Namely, it is required that noting should be leaked from the remotely stored files and index beyond the outcome and the pattern of search queries.

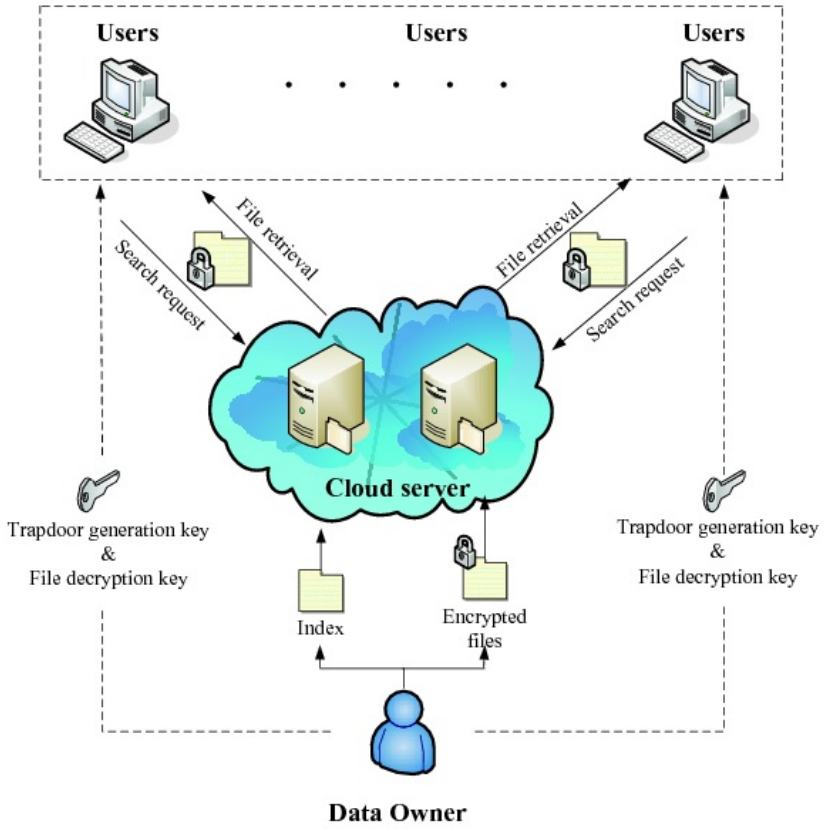


Figure 1: Architecture of verifiable fuzzy keyword search

In this paper, we construct a new verifiable fuzzy keyword search scheme, which can fill the gap between fuzzy keyword search and verifiability of searching result. To this end, we should achieve the following mainly goals: 1) to design storage-efficient index for fuzzy keyword sets; 2) to consturct efficient verifiable fuzzy keyword search scheme and enable user to verify the correctness and completeness of search result.

3 Constructions of Verifiable Fuzzy Keyword Search Scheme

In this section, we present the proposed verifiable fuzzy keyword search scheme in detail. What is need to point out is that the contents of files are separately encrypted by a symmetric encryption scheme in a conventional manner. We assume $SE = (Keygen, Enc, Dec)$ is symmetric encryption scheme, where $Keygen$ is setup algorithm with security parameter λ, Enc and Dec are the encryption and decryption algorithms, respectively.

For fuzzy keyword search, a trivial solution is the approach of listing all possible variants of keywords. By using this way, When constructing fuzzy keyword set $S_{w_i,d}$ for keyword w_i with edit distance d, we need to enumerate all possible words $w_i^{'}$ that satisfy $ed(w_i, w_i^{'}) \leq d$. It is clearly see that the list way introduces too many variants of keyword, which seriously influences the search efficiency.

3.1 Wildcard-based Fuzzy Set Construction

In the above list method, we observe that all the variants of the keywords must be listed even if only one operation is performed at the same position. To build a storage-efficient fuzzy keyword set, we utilize the wildcard technique proposed in [7] to construct fuzzy keyword sets. The idea is to consider the positions of the three primitive edit operations. Namely, we use a wildcard "$\star$" to denote all edit operations at the same position. The wildcard-based fuzzy keyword set of w_i with edit distance d is denoted as $S_{w_i,d} = \{S^{'}_{w_i,0}, \cdots, S^{'}_{w_i,d}\}$, where $S^{'}_{w_i,d}$ denotes the set of keywords $w^{'}_i$ 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 $S_{cat,1} = \{cat, \star cat, \star at, c\star at, c\star t, ca\star t, ca\star, cat\star\}$. compared to the list way, the total number of keyword variants reduces from 7*26+1 to 7+1 when edit distance $d = 1$.

3.2 The Proposed Construction

Based on the above construction of fuzzy keyword sets, we now present a description on the proposed verifiable fuzzy keyword search scheme over encrypted data. The proposed solution is describe as follows:

Initialization In this phase, the data owner prepares for data outsourcing. Firstly, he randomly generates the file encryption key k and uses it to encrypt his own files, then outsources them to the cloud server and receives the file identity ID_i. Secondly, the data owner randomly generates the trapdoor key tk [1]and shares it with all of the authority users.

IndexGen To achieve the goal of verification, all of the index items must be constructed into a concentrate data structure. That is, we must join all the discrete keyword trapdoors into a united index. In our solution, we utilize the well-known multi-way tree to store the fuzzy keyword set $w^{'}_{iw_i \in S_{w,d}}$ over a predefined symbol set. Note that all trapdoors have the same length of bit string. each one can be divided into l/n parts and each part can be denoted by n bits, where l is the the output length of one-way function $f(x)$. We assume $\Delta = \{\alpha_i\}$ is a predefined symbol set, where the number of different symbols is $|\Delta| = 2^n$ and each symbol $\alpha_i \in \Delta$ can be denoted by n bits.

In the multi-tree index, each node is arranged a specific Bloom filter as the $proof$. The Bloom filter of given node represents the current node and all of its children's prefixes [2]. This IndexGen phase consists of two steps.

- The data owner computes $T_{w^{'}_i} = f(sk, w^{'}_i)$ for each $w^{'}_i \in S_{w_i,d} (1 \leq i \leq p)$ with the trapdoor generation key tk. Then he divides the hash value into a sequence of symbols as $\alpha_{i_1}, \alpha_{i_2}, \cdots, \alpha_{i_{l/n}}$, where each symbol refers to n bits.

- The data owner builds a tree G_W covering all the fuzzy keywords $w_i \in W$. Initially, the root of the G_W denoted as Φ. For each trapdoor sequence, the first symbol is added into G_W as the child of the root, if there is no existing node equal to the symbol. Then a Bloom filter is computed for the root node, which represents the root node and all of the children of root. As a result, the root node stores a two-tuples (r_0, r_1), r_0 stores the symbol value; r_1 stores a bloom filter and its signature,[3]. Subsequently, the current node is moved to the node of the child of the root. The algorithm is carried out recursively. When the last symbol is performed, Different from the internal node, the

[1]The k and tk can be used the same one, we used different keys for higher security in this work.
[2]The prefix of each node refers to the sequence of symbol from root to current node.
[3]The signature refers to the MAC of the bloom filter.

$proof$ is the symbol sequence of the corresponding keyword and its signature. What's more,the corresponding identifier $FID_w||g_{tk}(FID_w)$ is attached the last node as a leaf node.

QueryGen When authorized user want to retrial some files contained specific keyword w, he creates firstly the corresponding fuzzy keyword set $S_{w,d}$ based on the above wildcard-based method [4]. Then he computes $T_{w'} = f(tk, w^{'})$ for each $w^{'} \in S_{w,d}$ as search query and sends $T_{w'}$ to the cloud server.

Note When $ed(w, w_i) < d$, $S_{w,d} \cap S_{w_i,d}$ must be not empty. For simplicity, that is because it is need at most d operations transform w to w_i and vice versa. The complete prove can be seen in [7]. Due to this result, we can find that the files contained the desirable keyword w must be retrieval back, if there exists a keyword w_i satisfied $ed(w, w_i) < d$.

Search Upon received the search request $T_{w'}$ from user, the cloud server is responsible for performing search based on the index. The phase includes the following steps:

- The cloud server transforms the search request $T_{w'}$ into sequence of symbols according to the predefined rule.
- The cloud server performs search for each trapdoor over the index. Specifically, For each trapdoor sequence, the first symbol is matched with the child of the root of G_W. If there is existing node equal to the symbol, set it as the current node. Subsequently, the second symbol is matched with the child of the current node. The algorithm is carried out recursively. When the current node is the leaf node, the search is over and returns the corresponding identifiers FID_w and the Bloom filter of the node as search result to the user, where the Bloom filter is $proof$. Note that the search will abort anytime in the process, if the match operation failed. In this case, only the Bloom filter of the current node will be return back as the $proof$.

Verify In this phase, the user checks the correctness and completeness of the searching result. The key idea is that the Bloom filter of node includes the children information of the current node by inserting the corresponding prefix into the Bloom filter. The phase includes the following steps:

- In the case of search successful, the user tests firstly the completeness of the searching result. Namely, he computes $g_{tk}(\hat{FID}_w)$ and tests whether $g_{tk}(\hat{FID}_w)$ is equal to the received $g_{tk}(FID_w)$, where $\hat{FID}_w$ is the concatenation of received identifiers. Similarly, the correctness of $proof$ can be checked. If pass, the searching result can be believed that it is complete, then he utilizes the $Proof$ to test the correctness of search result. The user can verifies the whether the $proof$ is honest because of the trapdoor generation key tk. That is, he reconstructs the signature with the tk.
- In the case of search failing, the user directly tests the correctness of searching result. Different from the case of searching successful, the user checks the correctness of Bloom filter of the current node. Due to the character of Bloom filter and the tree index structure, the user can check whether the desirable keyword consisted of the index. Namely, the user can check whether the prefix of the child of the current node in the index. Assume that r independent hash functions $h_1, \ldots, h_r$ used in the construction of Bloom filter. For example, the search request is $\alpha_2\alpha_1\alpha_4$ included in the index but the cloud declares that it cannot be find. So he return the Bloom filter of node $\alpha_1(\alpha_2)$ to the user. The user can find that the bloom filter contain 1's in all r locations for the $\alpha_2\alpha_1\alpha_4$. As a result, the user detect the cloud server is not honest.

[4] In fact, the edit distance of user's may be different from the predefined one.

4 Security analysis

In this section, we prove the security of the proposed verifiable fuzzy keyword search scheme.

Theorem 1. *The verifiable fuzzy keyword search scheme is secure regarding the search privacy.*

Proof. Similar to [7, 3], suppose the proposed scheme is not achieve the index privacy against the indistinguishability under the chosen keyword attack (IND-CPA), which means there exists an algorithm $\mathscr{A}$ who can get the underlying information of keyword from the index. Then, we build an algorithm $\mathscr{A}^{'}$ that utilizes $\mathscr{A}$ 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. $\mathscr{A}^{'}$ has an access to an oracle $O_{f'(\cdot)}$ that takes as input secret value x and return $f^{'}(x)$. Upon receiving any request of the index computation, $\mathscr{A}^{'}$ answers it with request to the oracle $O_{f'(\cdot)}$. After making these trapdoor queries, the adversary outputs two keywords w_0^* and w_1^* with the same length and edit distance, which can be relaxed by adding some redundant trapdoors. $\mathscr{A}^{'}$ picks one random $b \in \{0,1\}$ and sends w_b^* to the challenger. Then, $\mathscr{A}^{'}$ is given a challenge value y, which is either computed from a pseudo-random function $f(sk,\cdot)$ or a random function. $\mathscr{A}^{'}$ sends y back to $\mathscr{A}$, who answers with $b^{'} \in \{0,1\}$. Suppose $\mathscr{A}$ guesses b correctly with nonnegligible probability, which indicates that the value is not randomly computed. Then, $\mathscr{A}^{'}$ 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, $\mathscr{A}$ at best guesses b correctly with approximate probability 1/2. Thus, the search privacy is obtained.

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

Proof. To prove this, it is enough to find out an element in $S_{w,d} \bigcap S_{w_i,d}$. Since $ed(w,w_i) \leq d$, this means that at most d edit operations we can transform w to w_i. According to the definition of $S_{w_i,d}$, which is the set of all fuzzy keywords that has edit distance Less than or equal to d. It is clear that $w \in S_{w_i,d}$. Obviously, $w \in S_{w,d} \bigcap S_{w_i,d}$. That is, $S_{w,d} \bigcap S_{w_i,d}$ is not empty.

Theorem 2. *The verifiable fuzzy keyword search scheme is secure based on the verifiable fuzzy search.*

Proof. To prove the verifiability, we need to prove that the attacker can not forge a valid $proof$. To tamper the search result, the attacker need to forge the $proof$. There are three ways: (1) generate a r_1(Bloom filter) with different parameter; (2) randomly generate a r_1 to replace the original one; (3) return the r_1 of another node back to the user.

- For method (1) and (2), without the secret key pck, any attacker can construct a legal signature at most a negligible probability. That is, if the attacker attempt to return part of the search result or some fault ones, the attacker returns a forged $proof$. Due to the above reason, it will be detected by the private cloud.

- For method (3), assuming there are r independent hash functions $h_1,\ldots,h_r$ used in the construction of Bloom filter. Although it can pass the test of correctness of signature, the Bloom filter cannot contain 1's in all r locations for the query keyword. Namely, due to the query keyword w_q does not include in the Bloom filter,for the query keyword , all the r corresponding locations of $h_i(w_q)$ in the Bloom filter are 1 at most a negligible probability.

Baesd on the above analysis, without the secret key pck, the attacker can not construct a valid $proooof$. That is, our proposed scheme is secure based on the assumption of collision resistance of hash function.

Discussion Due to the probability of Bloom filter, there exists probability of falsely recognizing an unrelated keyword $w_i^{'}$. For a keyword w_i and its corresponding Bloom filter with a bit array of m bits, the probability of a false positive is about $f = (1-(1-/m)^{r|S_{w_i,d}|})^r \approx (1-e^{-r|S_{w_i,d}|/m})^r$. Although there exists false positive rate, the Bloom filter can detect the forged result with considerably high rate as well as very efficient computation and storage cost. As a result, it is a effective solution for the verification of keyword search.

5 Conclusion

In this paper, we investigated the fuzzy keyword search problem in the scenario of a semi-honest-but-curious server, which may execute only a fraction of the search and return part of the searching result honestly. We proposed a new efficient verifiable fuzzy keyword search scheme, which not only supports fuzzy keyword search over encrypted data, but also enjoys the verifiability of the searching result. Specially, our solution can efficiently detect whether the server is honest, and then prevent the stronger insider threat. Though rigorous security analysis, we showed that our method is secure and privacy-preserving, while correctly realizing the verifiable fuzzy keyword search.

Acknowledgements

This work is supported by the National Natural Science Foundation of China (No. 60970144 and 61100224), the Project Supported by Natural Science Basic Research Plan in Shaanxi Province of China (No. 2011JQ8042) and the Fundamental Research Funds for the Central Universities (Nos. K50510010003, JY10000901034 and K50510010030).

References

[1] Y. Lu and G. Tsudik, "Privacy-preserving cloud database querying," *Journal of Internet Services and Information Security (JISIS)*, vol. 1, no. 4, pp. 5–25, November 2011.

[2] Y. Shiraishi, M. Mohri, and Y. Fukuta, "A server-aided computation protocol revisited for confidentiality of cloud service," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 2, no. 2, pp. 83–94, June 2011.

[3] C. Wang, K. Ren, S. Yu, and K. Mahendra, "Achieving usable and privacy-assured similarity search over outsourced cloud data," in *Proc.of the 31th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'12), Orlando, USA*. IEEE, March 2012, pp. 451–459.

[4] 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*. IEEE, May 2000, pp. 44–55.

[5] R. Curtmola, 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*. ACM, October-November 2006, pp. 79–88.

[6] 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.

[7] 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*. IEEE, March 2010, pp. 1–5.

[8] 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*. IEEE, March-April 2009, pp. 604–615.

[9] S. Ji, G. Li, C. Li, and J. Feng, "Efficient interactive fuzzy keyword search," in *Proc. of the 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*. IEEE, April 2008, pp. 257–266.

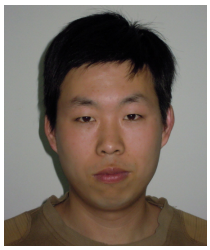
[11] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. of International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt'04), Interlaken, Switzerland, LNCS*, vol. 3027. Springer-Verlag, May 2004, pp. 506–522.

[12] 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*, vol. 4392. Springer-Verlag, February 2007, pp. 535–554.

[13] 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*. IEEE, June 2011, pp. 273–281.

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

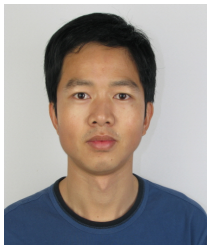
[15] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encypted data," in *Proc. of the 3rd Applied Cryptography and Network Security (ACNS'05), New York, NY, USA, LNCS*, vol. 3531. Springer-Verlag, June 2005, pp. 391–421.



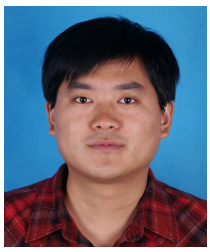
Jianfeng Wang is a gradute student at the Faculty of Science, in Xidian University. His research interests include public key cryptography, cloud computing security and searchable encryption.

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 in the Interdisciplinary Centre for Security, Reliability and Trust at University of Luxembourg. Before this, he was a postdoc researcher at the Distributed and Embedded Security Research Group in the Computer Science department at University of Twente, the Netherlands. 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 received his B.S. (2002) and M.S. (2004) from Southwest University and Sun Yat-sen University, both in Mathematics. He got his Ph.D degree in information security from Sun Yat-sen University at 2007. Currently, he works at Guangzhou University. His research interests include Applied Cryptography and Security in Cloud Computing (secure outsourcing computation and cloud storage). 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, IEEE Transaction on Information Forensics and Security, ESORICS etc. He also served as TPC committee for many international conferences on security. He received a National Science Foundation of China (NSFC) Grant for his research on secure outsourcing computation in cloud computing. He was selected as one of science and technology new stars in Guangdong province.



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.

Siqi Ma is a student at the School of Computer Science and Technology in Xidian University. Her research interests include cloud computing security and network Security.

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.