An Effective Search Scheme Based on Semantic Tree Over Encrypted Cloud Data Supporting Verifiability

Zhangjie Fu(✉), Jiangang Shu, and Xingming Sun

School of Computer and Software, Jiangsu Engineering Centre of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing 210044, China
{wwwfzj,kennethshu}@126.com, sunnudt@163.com

Abstract. With the increasing popularity of cloud computing, more and more sensitive or private information is being outsourced to cloud server. For protecting data privacy, sensitive data are always encrypted before being outsourced. Although the existing searchable encryption schemes enable users to search over encrypted data, these schemes support only exact keyword search, which greatly affects data usability. Moreover, these schemes do not support verifiability of search result. To tackle the challenge, a smart semantic search scheme is proposed in this paper, which returns not only the result of keyword-based exact match, but also the result of keyword-based semantic match. At the same time, the proposed scheme supports the verifiability of search result.

Keywords: Cloud computing · Semantic search · Verifiable search

1 Introduction

Cloud computing has become more and more prevalent, due to its benefits, including relief of the burden for storage, flexible data access, reduction of cost on hardware and software. More and more sensitive data (e.g. emails, personal health records and financial transactions, etc.) has been centralized into the cloud. It is common practice to encrypt sensitive information before outsourcing for protecting information privacy and alerting unauthorized access. However, data encryption makes existing search techniques on plaintext useless, thus prompting a big challenge to effective data utilization. A popular way to address this problem is searchable encryption, which can retrieve specific files through keyword-based search supporting data protection and keyword privacy-preserving.

In recent years, various efficient search schemes over encrypted cloud data based on searchable encryption have been proposed. However, these searchable encryption schemes based on keyword have two shortcomings. One is that most of these schemes support only exact keyword search. That means, the returned results are completely dependent on whether query terms users enter match pre-set keywords. The other one is that existing searchable encryption schemes assume that cloud server is honest-but-curious. However,

J. Tian et al. (Eds.): SecureComm 2014, Part I, LNICST 152, pp. 45–54, 2015.
DOI: 10.1007/978-3-319-23829-6_4

we noticed that cloud server may be selfish to save its computation or download bandwidth, which is significantly beyond the conventional honest-but-curious server model.

To meet the challenge of verifiable search and semantic search, in the paper, we propose an efficient verifiable keyword-based semantic search scheme. Our contributions in this paper can be summarized as follows:

(1) We propose a keyword-based semantic search scheme over encrypted data by building a semantic tree in real time, which can enable cloud server find out the keywords semantically similar to original query terms, improving the flexibility of system.
(2) By combining the keyword-based semantic search scheme with verifiable symmetric searchable encryption, we propose a search scheme supporting verification for search results. Our scheme is secure and privacy-preserving according to the rigorous security analysis.
(3) Our scheme is implemented and tested with real data sets. The extensive experiment results validate the practicality and efficiency of our proposed scheme.

2 Related Work

Symmetric Searchable Encryption: The first construction of symmetric searchable encryption (SSE) was proposed by Song et al. [1], in which after data encrypted symmetrically is outsourced into the untrusted server, client can search for data files by giving the server a search token that does not reveal any information on keyword or encrypted data. To achieve efficiency, Chang et al. [2] and Curtmola et al. [3] both build similar index, in which each entry contains encrypted trapdoor of a keyword and a series of corresponding encrypted file identifiers.

Asymmetric searchable encryption: Asymmetric searchable encryption (public-key version) [4] is used in a analogous scenario, except that anyone who owns the public key can encrypt and store data on a server, but only someone holding the private key can search and decrypt data files. Golle et al. [5], Hwang et al. [6] and Ballard et al. [7] have done some research on conjunctive keyword search. Then Boneh et al. [8] and Shi et al. [9] discussed some issues concerned with keyword conjunction and range query. Recently, great development on Asymmetric searchable encryption has achieved by some researchers.

However, all such research is focused on database field, not fully applied to cloud computing. To apply the searchable encryption to cloud computing, some researchers have been studying further on how to search over encrypted cloud data efficiently [10–14]. Li et al. [10] firstly proposed a fuzzy keyword search scheme over encrypted cloud data, which combines edit distance with wildcard-based technique to construct fuzzy keyword sets, to address problems of minor typos and format inconsistence. Wang et al. [11] proposed a secure ranked search scheme, in which through giving each keyword weight by TF-IDF, under the help of the order preserving symmetric encryption, the cloud server can rank relevant data files with no knowledge of specific keyword weigh. But this scheme supports only single keyword search. Then Cao et al. [12] proposed a privacy-preserving ranked scheme supporting multi-keyword, which

uses vector space model and characteristics of matrix to realize trapdoor unlinkablility and thereby preserves data privacy.

3 Preliminaries

Semantic Tree Model. In this paper, the semantic tree model to express the semantic relevance of the keywords will be constructed. In the model, the *m*-best tree is adopted as the semantic unit composed of keywords. When querying, the semantic tree will be set up in real-time based on query terms and some keywords satisfying the qualifications can be chosen out.

***m*-best Tree.** The *m*-best tree here is considered as the unit in the semantic tree model, which is composed of keywords. *sim(q,p)* denotes the similarity between word q and word p. Given any word q, a tree model can be used to express relationships with any other words. It is worth noting that all the leaf nodes from left to right are sorted according to the similarity with the root node. That means the more similar the leaf node is to the root node, the more left it will be. It should satisfy the following formula:

$$sim(q, p_1) \geq sim(q, p_2) \geq \cdots \geq sim(q, p_m) \tag{1}$$

sim(q,p) can be calculated by WordNet. The left most m leaf nodes are chosen and other leaf nodes are given up. It is called *m*-best tree, which is the unit in the semantic tree model. Note that the variable m can be adjusted according to the specific situation.

Term Similarity Tree. Given a query term vector $Q = (q_1,q_2,...,q_k)$ containing K terms, a term similarity tree *TST(Q,v,m)* based on Q in real-time can be built, as shown in Fig. 1.

The variable v is the number of layers in the tree and the variable m means each unit of the tree is the *m*-best tree. With the term similarity tree *TST(Q,v,m),* the similarity between the root node and any internal node or leaf node can be easily calculated. Specific definitions are described below:

(1) The path weight between the root node q_1 and the node p is multiply of all the weight in the path.
(2) The shortest path between the root node q_i and the node p is the path of the maximal weight.
(3) $sim(q_i,p)$ is the weight of the shortest path between them.

WordNet. WordNet is a lexical database for English language, which is created by Princeton University. It groups the English words into sets of synonyms called synsets, provides short, general definitions, and records the various semantic relations between these synonym sets. a variety of semantic similarity and relatedness measures based on WordNet can be easily implemented.

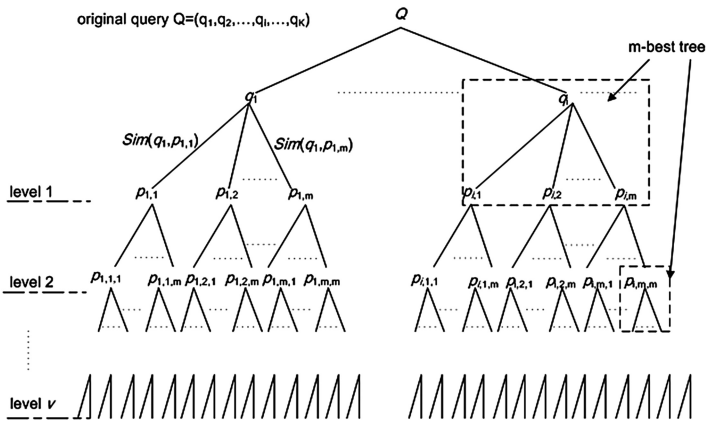

Fig. 1. TST(Q,v,m)

4 Verifiable Keyword-Based Semantic Search Scheme in Cloud

4.1 Technique for Keyword Semantic Extension Query

When the authorized user wants to retrieve some data files of his interest, he may type in some query terms, denoted as $Q = (q_1,q_2,...,q_k)$ which contains K query terms. To enable search more practical and flexible, a keyword extension technique to extend original query terms is proposed, getting some appropriate additional terms semantically related to the original query terms. Here, building the semantic similarity tree *TST (Q,v,m)* to extend original query terms Q is adopted. To extend query terms, firstly, *TST (Q,v,m)* should be built and then the extended terms meeting the requirements are chosen out. If the term v satisfies the following criteria, it can be considered as an extended term.

$$\begin{cases} sim(Q,w)=\sum_{i=1}^{K} sim(q_i,w) \geq K\times\varphi(v) \\ overlay(TST(Q,v,m),w) \geq \left\lfloor \frac{K}{2} \right\rfloor +1 \end{cases} \tag{2}$$

sim(Q,w) is the similarity between the original query terms Q and the term w. *sim(q_i,w)* is the similarity between the query term q_i and the term w. The $\varphi(v)$ is the increasing function of the level v, and it is defined as follows:

$$\varphi(v) = \sin(\frac{\pi v}{60}) + 0.5, v \in (0,10), v \in N \tag{3}$$

In the formula above, when $v \in (0,10), v \in N$, the domain of values of $\varphi(v)$ ranges from 0.5 to nearly 1. The value will increase with the increase of level v $overlay(TST(Q,v,m),w)$ denotes how many subtrees in *TST* the term w exists. $\lceil \frac{K}{2} \rceil$ will always round down to the nearest whole unit to $K/2$.

From the type, the first formula evaluates similarity between the term w and the query term $q_1,q_2,...,q_k$ and the second formula represents the similarity between the term w and the whole original query terms Q. The whole query terms Q represents user's semantic tendency of query, which can be seen as a point in the semantic space. The extended terms should be more similar to the point. That means, the extended term should be more similar to the whole query terms Q rather than one query term q_i among Q for the reason that the whole query terms Q expresses the explicit meaning, but one single query term q_i among Q cannot convey the explicit meaning.

4.2 The Verifiable Keyword-Based Semantic Search Scheme

In this section, the proposed scheme is emphatically presented in detail. The scheme includes five algorithms (**Setup, GenIndex, GenQuery, Search, and Verify**).

- **Setup**
  In this initialization phase, the data owner initiates the scheme to generate a random key $k \underset{\leftarrow}{R}\{0,1\}^k$ and a secret key $ZR\{0,1\}^L$.
- **GenIndex**
  Assuming that $\bullet = \{a_i\}$ is a predefined symbol set, where the number of distinct symbols is $|\bullet| = 2^{\bullet}$ and each symbol $a_i \in \Delta$ is denoted as $\bullet$-bit binary vector. Below, the L is the output length of the function $\pi(k, *)$.

  Preprocess:

  (1) The data owner scans the plaintext document collection D and extracts the distinct keywords of D, denoted as W;
  (2) The data owner computes the score S_{W_i,D_i} for each $W_i \in W$ and $D_j \in D$. For all data files containing the keyword W_i, the identifier set is denoted as $FID_{W_i} = ID(D_1) \,||\, \varepsilon_z(s_{W_i,D_1}), \ldots \,||\, ID(D_j) \,||\, \varepsilon_z(s_{W_i,D_j})$.

  Build the symbol-based index trie:

  (1) The data owner computes $T_{W_i} = \pi(k, w_i)$ for each $W_i \in W$ with the random key k, and then divides them into symbols as $T_{W_i} = \{a_{i,1}, a_{i,2}, \ldots, a_{i,L/\theta}\}$.
  (2) The data owner builds up a symbol-based index trie G covering all the T_{W_i} for each $W_i \in W$, where each node contains two attributes (r_0,r_1). r_0 stores the symbol in the $\bullet$; r_1 is a globally unique value $path||memory||g_k(path||memory)$ in G. The *path* is the sequence symbols from the root to the current node, denoted as $a_{i,1}, a_{i,2}, \ldots, a_{i,j}$, where $j \le L/\theta$; The memory is 2^{θ}-bit binary string, which represents the set of the children nodes of the current node. If the current node has a children node whose r_0 is the i-th symbol in $\bullet$, and then the i-th bit is set "1", while other bits are set "0". In parallel to build search index G, plaintext documents are separately encrypted by a symmetric way in a traditional manner.
  (3) The data owner attaches *IDSet* which is $\{FID_{W_i} \,||\, g_k(FID_{W_i})\}_{1 \le i \le n}$ to index G and outsources it together with encrypted files to the cloud server.

- **GenQuery**
  (1) When the user inputs the query terms $Q = (q_1,q_2,...,q_k)$, first builds term similarity tree *TST(Q,v,m)* and executes keyword semantic extension, getting the extended query $Q = (q_1,q_2,...,q_k,q_{k+1},...,q_m)$;
  (2) For each $q_i \in Q$, the user computes the trapdoor $T_{q_i} = \pi(k, q_i)$, and divides them into symbols as $T_{q_i} = \{a_{i,1}, a_{i,2}, \ldots, a_{i,L/\theta}\}$, finally sends $\{T_{q_i}\}_{q_i \in Q}$ to the cloud server. Meanwhile, the user should store $\{T_{q_i}\}_{q_i \in Q}$ temporarily to verify the search result later.

- **Search**
  Upon receiving the search request, the cloud server performs the search operation over the index *G*. The search is principally to find a path in *G* according to the search request, from the root node to the leaf node. The existence of a path indicates that the queried words happens at least one of the targeted data files. During every step of path exploration, the cloud server produces the *proof* which is later together with FIDS returned to the user for validity of search outcome. Note that the *proof* is the r_1 of each node found in the path during search, which is a globally unique value.

- **Verify and Rank**
  When the user receives the outcome from the cloud server, he can verify the correctness and completeness of search result. The key idea behind it is that the outcome returned by the cloud server contains the *proof*, which is a globally unique value and is produced by a pseudo-random function g_k with the random key *k*. Without the random key *k*, which is only shared in authorized users, the cloud server cannot forge a valid *proof*. The outcome returned by the cloud server can be divided into two situations: successful and unsuccessful.
  (1) If the outcome is successful, the outcome will contain IDSet and proof. Firstly, the user can verify the completeness of the IDSet, which consists of $\{FID_{W_i} \,||\, g_k(FID_{W_i})\}_{1 \le i \le n}$. The user can extract the FID_{W_i} and compute $g_k(FID_{W_i})$, where FID_{W_i} is the concatenation of identifiers received by the user. Then the user can test whether $g_k(FID_{W_i})$ is equal to the received $g_k(FID_{W_i})$. If they are equal, the user can consider the search result is complete. Otherwise, the search result is incomplete. After the first step, the user will utilize the proof to verify the correctness of the search outcome. Similar to the first step, the user computes the $g_k(path||memory)$ and tests whether it is equal to the received $g_k(path||memory)$, where the $path||memory$ is the former part of proof. If they are not equal, the user can see the cloud server is not worth being trusted.
  (2) If the outcome is unsuccessful, the user could directly verify the correctness of the search outcome. The proof is returned in the format of $G_{o,y_o}[r_1]||\ldots||G_{j,y_j}[r_1]\,||\,j$. b[j] is defined as a *j*-bit vector, where the last bit is set "0", other bits are set "1". This part of the process is to verify each unit $\{G_{j,y_j}[r_1]\}$ of proof, which contains three steps below:
    (a) The user computes $g_k(path||memory)$ and tests whether it is equal to received $g_k(path||memory)$, where $path||memory$ is the former part of the *proof*.

(b) If the first step pass, the user tests whether the received *path* is equal to corresponding $\{a_{i,1}, a_{i,2}, \ldots, a_{i,j}\}$ stored in user side.
(c) If the second step pass again, the user continue testing whether $memory[ord(T_{q_i}[j+1])]$ is equal to $b[j+1]$, where $T_{q_i}[j+1]$ is the next symbol of the current node according to the sequence symbol in the trapdoor. If not equal, the cloud server is lazy, that means, it only executed a fraction of search.

After verifying that the outcome is correct and complete, the user can decrypt IDSet with decryption key z and sort the returned data files.

5 Performance Analysis

Testing of Building Symbol-based Trie Tree. In the experiment, $\boldsymbol{\theta} = 4$ is chosen and SHA-1 is used as hash function with output length of L = 160 bits. So, the height of the symbol-based trie is $L/\theta = 40$ and that means every path in the trie is 40. Figure 2 shows the trie construction time. It shows that the construction time increases linearly with the number of distinct keywords. And the construction time is very fast and it can be conducted off-line and just one-time cost.

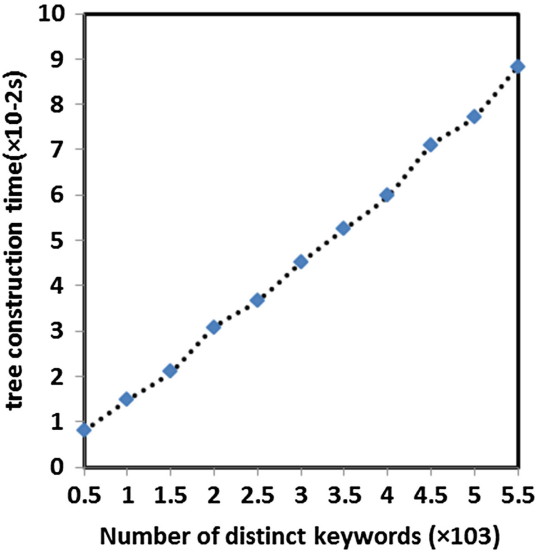

Fig. 2. Time cost to build trie

Testing of GenQuery. With the help of *m*-best tree, if users input a single query term, he will find its synonyms, various morphological forms and similar words. And when users input several query terms, he will find some words close to the whole query under the construction of term similarity tree. Therefore, our semantic search scheme supports

both single keyword and multi-keyword search. In our test, 20 users are invited to conduct a large quantity of queries to test the performance of keywordbased semantic extension. During the test, we continually adjust the variable m and v of the term similarity tree according to the feedback of users. Figure 3 shows the time cost to generate query of a single keyword.

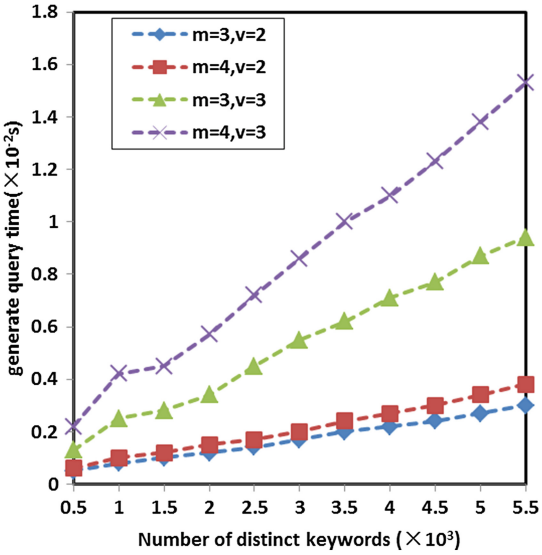

Fig. 3. Time cost to generate query

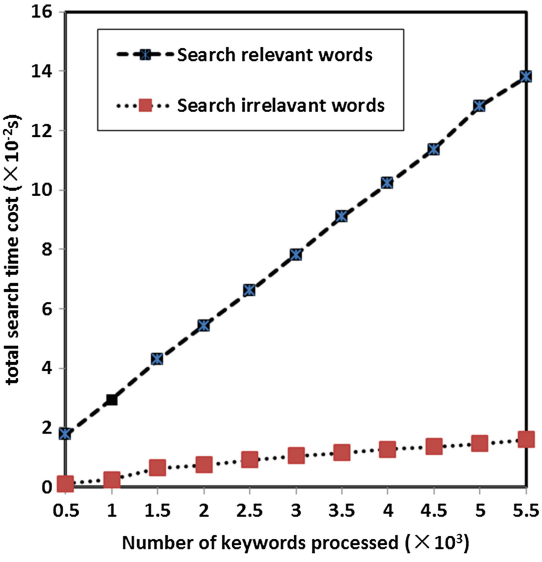

Fig. 4. Time cost to search

Testing of Search. Given the search index comprised of 5806 distinct keywords, we measure the time cost of search operation, shown in Fig. 4.

We obtained an estimation of throughput of search: 4000 words/second. In addition, we notice that searching for irrelevant word is much faster, which is because search will stop if there is a mismatch between symbols of trapdoor and the index. This "incomplete traversing" saves a lot of operating time.

6 Conclusion

In the paper, we propose an efficient verifiable keywordbased semantic search scheme. Comparing to most of the existing searchable encryption schemes, our scheme is more practical and flexible, better suiting users' different search intentions. Moreover, our scheme protects data privacy and supports verifiable searchability, in the presence of the semi-honest server in the cloud computing environment. Through ample theoretical analysis and experimental study using the real data set, our scheme is quite efficient.

Acknowledgements. This work is supported by the NSFC (61373133, 61232016, 61173141, 61173142, 61173136, 61103215, 61373132, 61272421), GYHY201206033, 201301030, 2013DFG12860, BC2013012, PAPD fund, Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology, Hunan province science and technology plan project fund (2012GK3120), the Scientific Research Fund of Hunan Provincial Education Department (10C0944), and the Prospective Research Project on Future Networks of Jiangsu Future Networks Innovation Institute (BY2013095-4-10).

References

1. Song, D., Wagner, D., Perrig, A.: Practical techniques for searches on encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy, pp. 44–55 (2000)
2. Chang, Y.-C., Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data. In: Ioannidis, J., Keromytis, A.D., Yung, M. (eds.) ACNS 2005. LNCS, vol. 3531, pp. 442–455. Springer, Heidelberg (2005)
3. Curtmola, R., Garay, J., Kamara, S., Ostrovsky, R.: Searchable symmetric encryption: improved definitions and efficient constructions. In: Proceedings of ACM Conference on Computer and Communications Security, pp. 79–88 (2006)
4. Boneh, D., Di Crescenzo, G., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 506–522. Springer, Heidelberg (2004)
5. Golle, P., Staddon, J., Waters, B.: Secure conjunctive keyword search over encrypted data. In: Jakobsson, M., Yung, M., Zhou, J. (eds.) ACNS 2004. LNCS, vol. 3089, pp. 31–45. Springer, Heidelberg (2004)
6. Hwang, Y.-H., Lee, P.J.: Public key encryption with conjunctive keyword search and its extension to a multi-user system. In: Takagi, T., Okamoto, T., Okamoto, E., Okamoto, T. (eds.) Pairing 2007. LNCS, vol. 4575, pp. 2–22. Springer, Heidelberg (2007)

7. Ballard, L., Kamara, S., Monrose, F.: Achieving efficient conjunctive keyword searches over encrypted data. In: Qing, S., Mao, W., López, J., Wang, G. (eds.) ICICS 2005. LNCS, vol. 3783, pp. 414–426. Springer, Heidelberg (2005)
8. Boneh, D., Waters, B.: Conjunctive, subset, and range queries on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 535–554. Springer, Heidelberg (2007)
9. Shi, E., Bethencourt, J., Chan, T.H., Song, D., Perrig, A.: Multi-dimensional range query over encrypted data. In: Proceedings of IEEE Symposium on Security and Privacy (SP 2007), pp. 350−364 (2007)
10. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.J.: Fuzzy keyword search over encrypted data in cloud computing. In: Proceedings of IEEE INFOCOM 2010, San Diego, CA, USA (2010)
11. Wang, C., Cao, N., Li, J., Ren, K., Lou, W.J.: Secure ranked keyword search over encrypted cloud data. In: Proceedings of IEEE 30th International Conference on Distributed Computing Systems (ICDCS), pp. 253–262 (2010)
12. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.J.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. Proc. IEEE INFOCOM **2011**, 829–837 (2011)
13. Chai, Q., Gong, G.: Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In: Proceedings of IEEE International Conference on Communications (ICC 2012), pp. 917–922 (2012)
14. Fu, Z., Sun, X., Liu, Q., Zhou, L., Shu, J.: Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing. IEICE Trans. Commun. **E98-B**(1), 190–200 (2015)
15. Zhang, H., Wu, Q.M., Nguyen, T.M., Sun, X.: Synthetic aperture radar image segmentation by modified student's t-mixture model. IEEE Trans. Geosci. Remote Sens. **52**(7), 4391–4403 (2014)
16. Li, J., Li, X., Yang, B., Sun, X.: Segmentation-based image copy-move forgery detection scheme. IEEE Trans. Inf. Forensics Secur. **10**, 507–518 (2015). doi:10.1109/TIFS.2014.2381872