Journal of Theoretical and Applied Information Technology
15st August 2011. Vol. 30 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SEARCHABLE SYMMETRIC ENCRYPTION: REVIEW AND EVALUATION

[1]YAP JOE EARN, [2]RAED ALSAQOUR, [3]MAHA ABDELHAQ, [4]TARIQ ABDULLAH

[1,2,3]Department of Computer Science, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600, Bangi, Selangor, Malaysia

[4]Faculty of Information Technology, Azal University for Science and Technology, Sana'a, Republic of Yemen

E-mail: joearn_yap@hotmail.com , raed.saqour@ftsm.ukm , maha@ftsm.ukm.my , dr.tariq@ieee.org

ABSTRACT

Searchable Symmetric Encryption (SSE) allows a user to search over their encrypted data on a third party storage provider privately. There are several existing SSE schemes have been proposed to achieve this goal. This paper concerns with three currentSSE schemes, which are the Practical Techniques for Searches in Encrypted Data (PTSED), the Secure Index(SI), and the Fuzzy Keyword Search over Encrypted Data in the Cloud Computing (FKS-EDCC).The objective of this paper is to introduce a review of the three schemes with a discussion in the advantages and disadvantages of each.This paper also implements aprototype over an SI-based secure file searching system using java language. The performance of the system has been evaluated and discussed according to the false-positive rate.

Keywords: *Searchable Symmetric Encryption (SSE), Practical Techniques for searches in encrypted Data (PTSED), Secure index (SI), Fuzz Keyword Search over Encrypted Data in Cloud Computing (FKS-EDCC).*

1. INTRODUCTION

Searchable Symmetric Encryption (SSE) [1] [2] allows a user to search over their encrypted data on a third party storage provider (server) privately. Nowadays, remote server is widely used by people to store data as a backup. In addition, people can access their remote data storage easily with their mobile devices or any other devices. For example, they can access both the mail server, and the history of the online chatting applications such as, mobile messenger and Facebook mobile chatting applications. SSE scheme is needed in this case to provide protection on privacy of the user.

Figure 1 illustrates the majority types of scenario happening in real life application, which needs the aid of searchable symmetric encryption. Client *C* wants to save some files into an un-trusted server *S*. These data later will be retrieved back by using other devices with limited computational power and capacity like the smart phone. Since server *S* is a remote server and cannot easily be-trusted, client *C* may need to encrypt the data and store into server *S*. However, client *C* may need to perform a search and retrieve only files that contain a certain keyword. In order to achieve the requirement, client *C* can retrieve all the files that he has saved in server *S*, then decrypt all files and search words by words. This searching method might take a very long time to complete since the device that the user used to perform the search has a limitation on capacity, computational power and also bandwidth range.

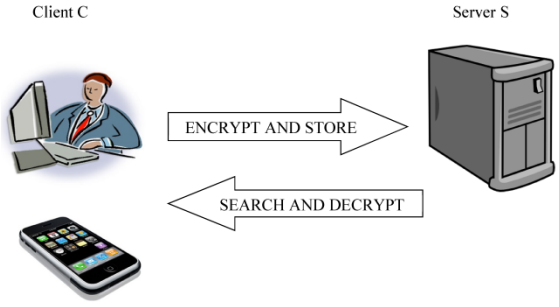

Figure. 1 Searchable Symmetric Encryption

In this paper, we reviewed three of the current existing SSE schemes. The first scheme is Practical

Journal of Theoretical and Applied Information Technology
15st August 2011. Vol. 30 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Techniques for Searches in Encrypted Data (PTSED) proposed by Song et al. [1]. The second scheme is Secure Indexes (SI) proposed by Goh[2]. And the third scheme is Fuzzy Keyword Search over Encrypted Data in Cloud Computing (FKS-EDCC)proposed by Jin Li. et al. [3].The three SSE schemes were selected on this study since there are some similarities between them in several aspects. Firstly, all of the schemes include a number of prebuilt functions to hide the information of the searching keyword form. Secondly, three of these schemes require a high performance machine to compute the complex algorithms for keyword building and searching at the server but not for client retrieving devices. Thirdly, all of the SSE schemes also add a few more enhancements such as the occurrence frequency search for the keyword. This occurrence frequency approach can be applied by attaching a few more information bits with each of the keywords.

To the best of our knowledge, this study is the first to compare the three schemes relating to their advantages and disadvantages. In addition, it developed a prototype of a secure file searching by make use of SI scheme and evaluated the performance of the scheme in terms of false-positive rate and execution time for different file sizes. The main part of the whole scheme is searching for the keyword from an encrypted file.

This paper is structured as follows: Section 2 introduces the related works, PTSED, SI, and FKS-EDCC scheme.Section 3presentsthe comparison between the three schemes. Section 4 introduces the results and discussion. And Section 5 concludes the paper and shows the future work.

2. RELATED WORKS

2.1 Practical Techniques for Searches in Encrypted Data (PTSED) scheme

The PTSED scheme [1] was proposed by Song et al. The scheme is based on sequential scan method. Before the authors proposed the PTSED scheme, they started from a very basic encrypting and searching scheme. They first started with a basic scheme and show that its encryption algorithm provides provable secrecy. Then they show how the scheme can be extended to handle controlled searching and hidden searches [1]. After considering a few criteria and scheme modifications, the authors come out with a complete version of PTSEDscheme.

PTSED consists of several steps: Pre-encryption, searching, and decryption. The purpose of the pre-encryption first step is to hide the actual searching keyword and to prevent any unauthorized party which can excess the remote server using cryptanalysis to break the whole encrypted message after a few keyword searches.Before starting the searching algorithm, the user has to provide some information sincethe server will not learn anything more than what is provided by the user. After the server gathers the required information from the user, the searching algorithm will run based on the information gathered. In this case, the server may return the file to the end user if the keyword is match. Otherwise, it will continue to search until the end of the file. After the user search and retrieve the encrypted file containing the specific keyword, the final step is to decrypt the retrieved file back to plaintext.

2.2 Secure Index(SI) scheme

The SI scheme [2] was proposed by Goh. The scheme builds a secure index for documents. This secure index allowsa user to search for an encrypted document that iscontaining a keyword without decrypting the document. A Bloom Filter (BF)[4-5] is used as a per document index to keep track of each of the unique words. Before each of the unique keywords is indexed and stored into bloom filter objects, those unique keywords have to go through a pseudorandom function twice. The purpose of doing so is to make sure that for each two or more documents, if they contain the same keyword the codeword will represent it differently.

SI scheme consists of the following four algorithms:

- *Keygen (s)*: Given security parameter *s* to a secure pseudorandom function, the function generates a master private key$K_{priv} = (k_1, \dots\dots, k_r)$. Where *r* represents the number of sub-key. For example, if the total length for K_{priv} is 100 bits and the number of *r* is 10, theneach sub-key is 10 bits long.
- *Trapdoor*(K_{priv}, w)*:* Given a word *w* and the keyK_{priv}, outputs a trapdoor for word *w* as $T_w = (f(k_1, w), f(k_2, w), \dots\dots, f(k_r, w))$ where $f(.)$ is the key hash message authentication function (e.g. HMAC-SHA1). The trapdoor function hides the information of the original word *w* while performing searching process and hence strengthens the security level for the secure

Journal of Theoretical and Applied Information Technology
15st August 2011. Vol. 30 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

index scheme. This algorithm consumesO(1) time to produce the trapdoor.

- *BuildIndex*(D, K_{priv}): Input a document D with a unique identifier (name of the document) D_{id}, and the private key K_{priv} , this function identifies the set of unique word $(w_1, w_2, \dots\dots, w_t)$ in a document D. The algorithm performs the following computations:

  a) For each unique word w_i, for $i \in [0, t]$, compute:
  - The trapdoor$\left(x_1 = f(k_1, w_i), x_2 = fk2,wi, \dots\dots, xr=fkr,wi.\right.$
  - The codeword $\left(y_1 = f(x_1, D_{id}), y_2 = fx2,Did, \dots\dots, yr=fxr,Did.\right.$
  - Hashes the codeword into a documentD_{id}'s BF.

  b) Output the index of the input document$I_{D_{id}} = (D_{id}, BF)$.

This build index algorithm take linear time in the numbers of words contain by document *D*.

- *SearchIndex*$(T_w, I_{D_{id}})$: Given the trapdoor for a keyword *w*, $T_w = (x_1, x_2, \dots\dots, x_r)$ and the index $I_{D_{id}} = (D_{id}, BF)$. This algorithm performs the following computations:

  a) Computes codeword $(y_1, y_2, \dots\dots, y_r)$ for the keyword y by taking D_{id} and T_w going through the HMAC-SHA1 pseudorandom function.
  b) Hashes and checks the position bits of this codeword in BF.
  c) Returns 1 if all the position of codeword y is set. Otherwise returns 0 to indicate fail on matching.

2.3 Fuzzy Keyword Search over Encrypted Data in Cloud Computing (FKS-EDCC) scheme

FKS-EDCC scheme [3] was proposed by Jin Li. et al. The scheme focuses on enabling effective yet privacy preserving fuzzy keyword search in cloud computing [3]. Fuzzy keyword search returns the matching files when the users search the inputs which exactly match the predefined keywords. If the match fails, the closest possible matching files based on keyword similarity semantics is returned.

Straightforward approach is first proposed to achieve all the functions of fuzzy keyword search. It shows how fuzzy search scheme works over encrypted data. The scheme begins by constructing the fuzzy keyword set $S_{wi,d}$ for each keyword $w_i \in W(1 \le i \le p)$ with edit distanced which is used to quantify keyword's similarity. 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 $ed(w_i, w'_i) \le d$, that is, all the words with edit distance d from w_i are listed [3].

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 FID_{w_i}as $Enc(sk, FID_{w_i}||w_i)$. The index table $\{(\{T_{w'_i}\}_{w'_i \in S_{wi,d}}, Enc(sk, FID_{w_i}||w_i))\}_{w_i \in W}$ and encrypted data files are outsourced to the cloud server for storage.
2. To search with w, 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 $\{Enc(sk, FID_{w_i}||w_i)\}$ according to the fuzzy keyword definition. The user decrypts the returned results and retrieves relevant files of interest.

Based on the straightforward approach, all the variants of keywords need to be listed downeven if an operation is performed at the same position. As a result of that weakness, Jin Li. et al. propose to use a wildcard to denote edit operation at the same position. The wildcard-based fuzzy set of w_i with edit distance d is denoted as $S_{w_i,d} = \{S'_{w_i,0}, S'_{w_i,1}, \dots, S'_{w_i,d}\}$, where $S'_{w_i,T}$ denotes the set of words w_i with T wildcards. Each wildcard represents an edit operation on w_i.

The scheme of fuzzy keyword search is as follows:

1. To build an index for w_i with the edit distance d, the data owner first constructs a fuzzy keyword set $S_{w_i,d}$ using the wildcard based technique. Then 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 also encrypts FID_{w_i} as $Enc(sk, FID_{w_i}||w_i)$.The index

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

table $\{(\{T_{w'_i}\}_{w'_i \in S_{wi,d}}, Enc(sk, FID_{w_i}||w_i))\}_{w_i \in W}$ 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. 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 $\{Enc(sk, FID_{w_i}||w_i)\}$ according to the fuzzy keyword definition. The user decrypts the returned results and retrieves relevant files of interest.

The technique of constructing the 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. By this way, the searching result correctness can be ensured.

3. COMPARISON

The main difference between all of the three schemes is that PTSED scheme uses a sequential approach. Sequential approach works well for a small size data, but it will be the worst algorithm when the size of the file gets larger and larger. The worst case happens when the targeted keyword is located at the end of file or the file does not contain the keyword which is being searched. For the searching algorithm, the server has to generate the sub-key for each block. If the worst case happens for a very large file, the performance and efficiency of the server will be degraded as it needs to perform a lot of key computations without getting any outputs.

In the SI scheme, the idea is that this scheme will perform a filter to make sure that each of the keywords is unique and non-duplicated before storing into the bloom filter. Assume that increasing in file size will increase the number of unique keywords, the larger the file size, the larger the bit vector size is needed to maintain the false-positive rates for the bloom filter. This will only affect the size of the bloom filter objects, the unique keyword filtering process and also the capacity of the server. The efficiency of the searching algorithms will not be affected since the searching for elements in the bloom filter is only depending on the number of hash functions in the bloom filter k. The k value can be maintained by fixing the ratio of bit vector size and the expected number of elements (keywords) $\frac{m}{n}$. The other advantage for this scheme is that it only required *O(1)* time to compute the private key and the trapdoor [2]. This private key will then be used by all the unique keywords to generate a unique trapdoor and codeword without reconstruction. Hence, it can improve the overall performance and efficiency of the machine while running the building and searching algorithms.

The most difference in FKS-EDCC scheme is that most of the existing searchable encryption techniques do not suit for cloud computing scenario, since they support only exact keyword search. That is common if the users' input is not matched exactly with those pre-set keywords due to some possible typos or wrong spelling and format inconsistencies. Fuzzy keyword search is supported via a simple spell check mechanism.

The comparison between PTSED, SI, and FKS-EDCC schemes is summarized in Table 1.

Table. 1A comparison between PTSED, SI, and FKS-EDCC schemes

Characteristics	PTSED scheme	SI scheme	FKS-EDCC scheme
Hide the information of the searching keyword form.	Yes	Yes	Yes
Require a high performance machine to compute the complex algorithms	Yes	Yes	Yes
Sequential approach	Yes	No	No
Document index approach	No	Yes	No
Perform filter (keywords is unique and non-duplicated)	No	Yes	No
Fuzzy keyword approach	No	No	Yes
Cloud computing	No	No	Yes

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4. RESULTS AND DISCUSSION

After comparing between all of the mentioned schemes, SI scheme is chosen to be used in this paper since this scheme have the advantage among the mentioned schemes, to deal with larger file size without decreasing the performance and efficiency of the machine as in the most cases. This section displays the results gained from the SI implementation.

4.1 Testing Files

Four text files have been created as a sample to test on the performance of SI scheme. *ID.txt* filecontains 1000 student ID. Mark.txt file contains 1000 random numbers ranging from 0-100. Name.txt file contains 550 names of students, and the last file *Articles.txt*contains a complete online article [1]. The first entry for *ID.txt*, *Mark.txt* and *Name.txt* indicates the size of data contains in each file respectively. The testing scenario is that the client wants to find out the files whichcontain 1000 entries.

4.2 False-positive rate for the bloom filter object

Table 2 shows the false-positive rate for the different ratios of *m/n* and the number of hash functions*k*. As shown in figure 2, it can be observed that the false-positive rate decreases exponentially with the increase ration of *m/n*. when the ratio gets closer to20, the false-positive rate would be negligible. The constructor for the bloom filter sets the bits' vector size for bloom filter to 20000 and the numbers of the predicted input data to 2000. The number of hash functions that is generated is 7.

Table. 2 False-positive rates

Ratio m/n	k	False-positive rate (%)
0	0	100
1	1	63.31
2	1	39.35
3	2	23.68
4	3	14.69
5	3	9.18
6	4	5.61
7	5	3.47
8	6	2.16
9	6	1.33
10	7	0.82
11	8	0.31
12	8	0.19
13	9	0.19
14	10	0.12
15	10	0.07
16	11	0.05
17	12	0.03
18	12	0.02
19	13	0.01
20	14	0.01

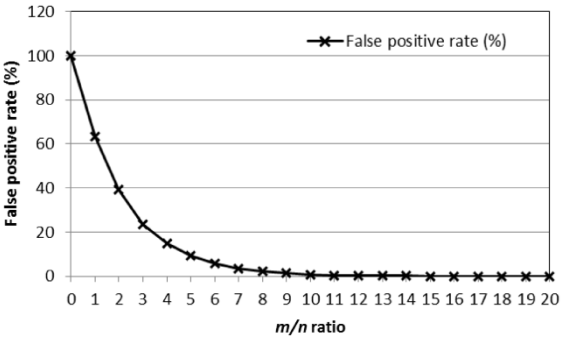

Figure.2False-positive rate (%) versus m/n ratio

4.3 Average Execution Time

The results which are shown in this paperis the average values for five samples. Thecodes for calculating the execution time is shown in Figure 3.

```
----------Testing Modules----------
Date StartTime = new Date();
Date EndTime = new Date();
long ExeTime = EndTime.getTime()-
StartTime.getTime();
```

Figure.3The testing Method

The Date objects get the current time from the running machine once the empty constructor is called. *getTime()* functions which are provided by this Date object return the time that is stored in the object in milliseconds. This is one of the builds in the class which is provided by Java API.

Figure 4, Figure 5, and Figure 6 depict the execution time for *buildIndex()* algorithm of SI, *Trapdoor()* algorithm of SI, and *SearchIndex()* algorithm of SI respectively. The results wereobtained by skipping all the user interface processes such as file selecting and keyword inserting. Hence, the programhas a longer execution time in real application if compared to the sum of execution time in each figure below.

The execution times for some algorithms are highly dependent on the size of the input file. The larger the size of the file or the more the content in

Journal of Theoretical and Applied Information Technology
15st August 2011. Vol. 30 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the file, the longer the time is needed to run these algorithms. This kind of situation can be figured out through Figure 3. *ID.txt* file needs longer execution time followed by *Articles.txt*, and then *Name.txt*, and the final one is*Mark.txt*.

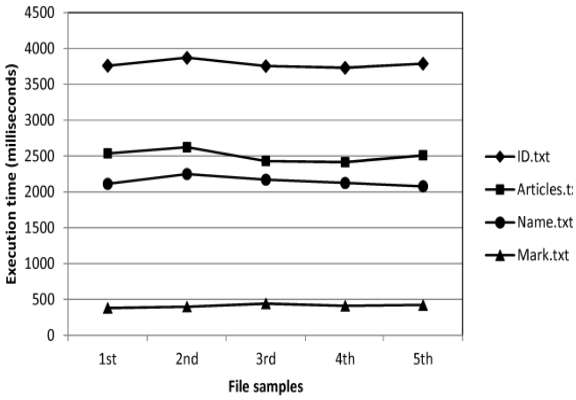

Figure.4The execution Time of the buildIndex() versus the total of 5 file samples

The *Trapdoor()*algorithm is taking only one argument, which will be the keyword wanted to search for, without performing the codeword computation process. This meansthat the execution time depends on the keyword size. This is the reason why for each testing, the execution time will be slightly different. The execution time in Figure 4 is in the range from 270ms to 305ms.

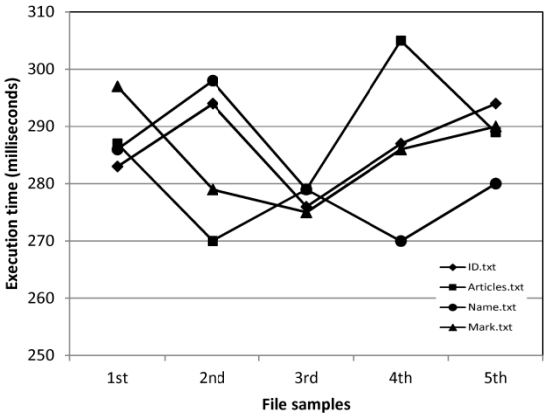

*Figure.5The execution time of the Trapdoor()*algorithm *versus the total of 5 file samples*

The *searchIndex()*algorithm needs to compute the codeword from trapdoor value containing in *Trap.dat* with the unique file identifier which obtained from *Directory.dat* file. After the codeword computation is done, compare the codeword generated with the respective bloom filter object from *Filter.dat*. The execution time depends on the comparison between codeword and bloom filter object. This is the reason why for each testing, the execution time will be slightly different. The execution time in Figure 5 is in range from 380ms to 403ms.

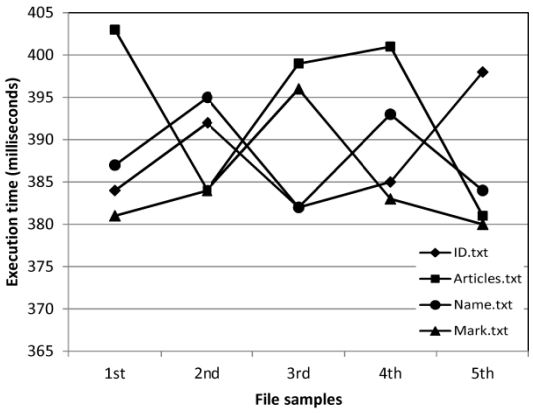

*Figure.6 Execution time of searchIndex()*algorithm *versus total of five file samples*

5. CONCLUSION AND FUTURE WORKS

In this paper, we studied three of the current existing SSE schemes, which are Practical Techniques for Searches in Encrypted Data proposed by Song et al. in the year 2000, Secure Index proposed by Goh in year 2004, and Fuzzy Keyword Search over Encrypted Data in Cloud Computing proposed by Jin Li. et al. in year 2010.

Based on the literature review, Goh's scheme is more preferable to build among Song et al.'s scheme and Jin Li. et al.'s SSE scheme since it has the capability to deal with larger file size more efficiently.

REFRENCES:

[1]. Song, D., Wagner, D., and Perrig, A. (2000), "Practical techniques for searching on encrypted data," In Proceedings of 2000 IEEE Symposium on Security and Privacy, pp. 44–55, May 2000, doi: 10.1109/SECPRI.2000.848445

[2]. Goh, E-J., (2004), "Secure indexes," in Cryptology ePrint Archive: Report 2003/216, February 25, 2004.

[3]. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., and Lou, W. (2010), "Fuzzy keyword search over encrypted data in cloud computing," In Proceedings of IEEE

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

INFOCOM'10 Mini-Conference, pp. 1-5, doi: 10.1109/INFCOM.2010.5462196

[4]. Bloom, B. (1970), "Space/time trade-offs in hash coding with allowable errors," Communications of ACM, vol 13, issue 7, pp. 422-426.

[5]. Marais, J., and Bharat, K., (1997), "Supporting cooperative and personal surfing with a desktop assistant," In Proceedings of ACM UIST'97, October 1997.

AUTHOR PROFILES:

Yap Joe Earnis a graduate student at National University of Malaysia.She expected to graduatefrom National University of Malaysia with a bachelor degree in computer science and information technology in September 2011.She is now working as a junior web programmer at a company in Singapore with senior guide.

Raed Saqouris a senior lecturer in the faculty of computer science & information technology at Universiti Kebangsaan Malaysia. Hereceived his B.Sc. degree in computer science from Mu'tah University, Jordan, M.Sc. degree in distributed system from University Putra Malaysia, Malaysia, and his PhD degree in wireless communication system from Universiti Kebangsaan Malaysia, Malaysia. His research interests include wireless network, ad hoc network, routing protocols, and network performance evaluation.

Maha Abdelhaq received her B.Sc. degree in computer science from Jordan University, Jordan,M.Sc. degree in computer science from Jordan University, Jordan. She expected to receive her PhD degree in wireless ad hoc network security in December 2012 from Universiti Kebangsaan Malaysia, Malaysia. Her research interests include ad hoc network, routing protocols, network security, artificial computational intelligence and network performance evaluation.

Tariq Abdullah is an assistance professor in faculty of computer science at Universiti Azal University for Science and Technology. Yemen. He received his B.Sc. degree in computer science from Baghdad University, Iraq, M.Sc. degree in distributed computing from University Putra Malaysia, Malaysia, and his PhD degree incomputer networks and protocol designfrom University Putra Malaysia, Malaysia. His research interests include wireless networks, protocols design, and network performance evaluation.