Wireless Personal Communications (2021) 116:3119–3142
https://doi.org/10.1007/s11277-020-07839-5

A New Searchable Encryption Scheme with Integrity Preservation Property

Mohammad Zamani[1] · Masoumeh Safkhani[1] · Negin Daneshpour[1] · Amir Abbasian[1]

Accepted: 14 September 2020 / Published online: 19 October 2020
© Springer Science+Business Media, LLC, part of Springer Nature 2020

Abstract

Searchable encryption schemes allow documents' owners to store their encrypted documents on servers, search for the desired keyword and then download only the desired encrypted file and then decrypt. Storing files on remote servers can be further developed, which, in addition to allow access to files at any location and at any time, it also gives the data owners the confidence that their files are stored without any change in the servers. This property can be interpreted as the integrity preservation property of encrypted documents in the searchable encryption schemes. Recently, in Yang et al. (Concurr Comput Pract Exp 29:e4211, 2017), Yang et al. proposed a semantic keyword searchable proxy re-encryption scheme and claimed their scheme resists against collusion and provides data privacy. However, in this paper, we show that unfortunately, their searchable encryption scheme does not provide the perfect security and their scheme is vulnerable against integrity contradiction attack. Our proposed attack is implemented in three different scenarios, its success probability of each scenario is one and its complexity is only one run of the scheme. We also improve Yang et al. scheme and show informally and formally that the improved scheme is secure against the attacks presented in this paper and also other known active and passive attacks. Comparisons also showed that the proposed scheme, in addition to the complete security it provides, is acceptable in terms of communication, storage and computational costs.

Keywords Searchable encryption scheme · Encrypted documents · Digital signature · Integrity preservation · Integrity contradiction attack · Cloud computing

✉ Masoumeh Safkhani
safkhani@srttu.edu

Mohammad Zamani
m.zamani@srttu.edu

Negin Daneshpour
ndaneshpour@srttu.edu

Amir Abbasian
a.abbasian@srttu.edu

[1] Computer Engineering Department, Shahid Rajaee Teacher Training University, Tehran, Iran

1 Introduction

One of today's needs is remote storage resources that can be accessed comprehensively and from everywhere, such as Gmail servers, Yahoo mail, and etc. Users typically encrypt sensitive data on honest-but-curious or semi honest-but-curious servers. Encryption hides all information about the data, and the client must download and decrypt all encrypted documents so that he/she can find the document with the specific keyword. Searchable encryption schemes help the client to only download and then decrypt the specific document of target. In fact, searchable encryption schemes attempt to help the client searchs its document among encrypted files by disclosing minimum information to the server. At the same time, an issue that is often neglected is the integrity preservation of the encrypted documents during the transfer to the server or vice versa. If there is not any protection from encrypted documents, then the data owner or authorized client will never reach the correct and original document. Suppose these documents are related to the hospital patients. If this encrypted information is changed during the transfer to the server for storage or during the transfer to the data owner or the client access after searching a keyword, without the server or data owner or client realizing, it can lead to irreparable risks. For an example, changing patient test results may lead to misdiagnosis, which can have serious complications for the patient. So far, numerous searchable encryption schemes have been proposed [10,13,14,19,20,22,25,26,29,30,37,40,43,44,46]. The security analyzes that have been made afterward on the schemes such as [1,15,17,18, 21,23,24,31,35,38] have shown that they have not been able to fully achieve their security goals. In this paper, we analyze and improve a recently proposed protocol by Yang et al. [42].

1.1 Problem Statement and Our Contributions

Recently, Yang et al. [42] proposed a new semantic keyword searchable proxy re-encryption based on lattice and claimed their scheme satisfies documents and keywords privacy. However, in this paper, we show that their scheme is vulnerable against integrity contradiction attack. Our attack can be implemented in three different scenarios. In any scenario, the probability of success is equal to one and its complexity is only one run of the scheme. In addition, we improve their scheme and show that the improved scheme is resistant against the attack proposed in this paper and also other known active and passive attacks. Comparing the proposed scheme with other related schemes shows that in addition to providing complete security, it also has acceptable computational, communication and storage costs.

1.2 Related Work

In general, the related work in this paper is divided into three categories of searchable encryption, proxy re-encryption and proxy re-encryption with keyword search that we will explain in each case.

Searchable Encryption Searchable Encryption is the best encryption for cloud servers or databases due to lower cost of data decryption on the server and searchable data. In 2000, Song et al. [34] proposed the first symmetric searchable encryption scheme in which the private key cryptographic structure was used. In the following years, other schemes were presented (see [9,12,16]), but the problem was that these structures were suitable for single writer/single reader (S/S) architecture and were not well structured for multi writer/single reader (M/S) projects due to the high cost of building secure channels for private key transfer. In 2004, Boneh et al. [7] provided the first public key searchable encryption scheme called Public key Encryption with Keyword Search (PEKS). Later, a lot of schemes were presented in the field of PEKS that referred to Ma et al.'s scheme [27] (called secure channel free certificateless searchable public key encryption with multiple keywords) that was presented for use in Internet of Things (IoT). The scheme used public channels to transmit messages, but the scheme was vulnerable to keyword guessing attack. In 2018, Wu et al. [37] presented a scheme which, in addition to resistance to the keyword guessing attack, was resistant against file injection attack.

Proxy Re-Encryption Proxy Re-Encryption (PRE) concept, which was introduced by Blaze, Bleumer and Strauss [6] in 1998, refers to that a third party (the same proxy server) converts the data owner encrypted document into the data user encrypted document using his public key, while a proxy server must not obtain any useful and valuable information. This scheme was bidirectional PRE, which means that changing the encrypted document into the opposite direction (i.e., the data user to the data owner) is possible and resistant to the chosen-plaintext attack. In 2006, Ateniese et al. [3] proposed the first unidirectional scheme, which used the basics of cryptography bilinear pairings, and in addition to being safe against chosen-plaintext attack, it had a faster and more efficient structure. The next design created by Canetti and Hohenberger [8] was based on Decisional Bilinear Diffie–Hellman in standard model, which was secure under chosen-ciphertext attack (CCA). Shao and Cao's scheme [32] was also resistant to collusion, in addition to being secure against the CCA. This was the first scheme of unidirectional PRE proposed schemes with CCA and collusion-resistance features.

Proxy Re-Encryption with keyword search In 2010, Shao et al. [33] introduced a new concept called "proxy re-encryption with keyword search", in which the user creates a trapdoor using a private key, then by proxy server searches it in the ciphertext without any other information from the proxy server. Since the scheme of [33] suffered from a single keyword restriction, Wang et al. [36] proposed a scheme to address this limitation. Shao et al. [33] and Wang et al. [36] proved their schemes in the random oracle model. In next scheme, Yang et al. [41] proposed a scheme called conjunctive keyword search with designated tester and timing enabled proxy re-encryption function (Re-dtPECK), which gives the user time-dependent encryption scheme that allows the user to access the searching for data and decrypt the encrypted document only at that time period. In their proposed scheme, only a designated server will be able to search query keywords, which makes it safe for keyword guessing attack. Obviously, these time-dependent schemes may not meet all the security needs. In 2017, Yang et al. [42] proposed an unidirectional encryption scheme (i.e. only can convert a delegator ciphertext into the ciphertext for delegatee but not in the opposite direction), suggesting that it is resistant to attacks on quantum computing and computers. Another feature of the scheme was that it was able to search for synonym keywords with the keyword, in addition to searching for a keyword. But their scheme had a drawback that it was not resistant to the integrity contradiction attack. Xu et al. [39] proposed another scheme called time controlled public key encryption with delegated conjunctive keyword search, tc-PEDCKS, which provided the ability to search for conjunctive keywords. This scheme is able to share data of multiple data owners for one data user and provides a certain amount of time to access the data of other data users.

Table 1 Notations used in this paper

Notation	Description
s	A cloud server
p	A proxy server
o	A data owner or delegator
c	An authorized client or delegatee
A	Adversary
TTP	The trusted third party
k	A security parameter which returns pk_i and sk_i
pk_i	A public key of user i
sk_i	A secret key of user i
$rk_{i\to j}$	A re-encryption key
KW	A keyword
$T_{KW,i}$	A trapdoor of keyword KW for user i
Enc	Encryption function
h	An one-way hash function
EF_i	An encrypted file of user i which is computed as $Enc(pk_i, file)$
$file$	A plain form of user i's document
CT_i	A secure index which is the encrypted of keyword KW for user i and computed as $Enc(pk_i, KW)$
$KeyGen(k) \to (pk_i, sk_i)$	A function that receives k and outputs (pk_i, sk_i)
$ReKeyGen(sk_i, pk_i, pk_j) \to (rk_{i\to j})$	A function that receives (pk_i, sk_i) of user i and the public key of user j and outputs $rk_{i\to j}$
$Trapdoor(sk_i, pk_i, KW) \to (T_{KW,i})$	A function that receives (pk_i, sk_i) and KW of user i and outputs $T_{KW,i}$
$Enc(pk_i, KW) \to CT_i$	A function that receives the public key of user i, pk_i and a keyword $KW \in \{0, 1\}^*$ and outputs CT_i
$ReEnc(rk_{i\to j}, CT_i) \to CT_j$	A function that receives the re-encryption key $rk_{i\to j}$ and the CT_i of user i and outputs CT_j for user j
$Test(pk, CT, T_{KW}) \to \{1\ or\ 0\}$	A function that receives pk, CT and T_{KW} and outputs 1 if CT, includes KW otherwise it outputs 0
$Sign(sk_i, m) = S$	A digital signature function which each entity i can sign m by using its secret key sk_i
$DeSign(pk_i, S) = m$	A function that anyone with a signature and a public key of the signer can verify the signed message

1.3 Organization

The rest of the paper is organized as follows: Sect. 2 is dedicated to the description of Yang et al. searchable encryption scheme. Our proposed integrity contradiction attack is applied on the scheme in Sect. 3. In Sects. 4 and 5, we describe the improved searchable encryption scheme and its security analysis respectively. We also compare the proposed scheme with other related schemes in computational and communication aspects of views in Sect. 7. Finally, the paper concludes in Sect. 8.

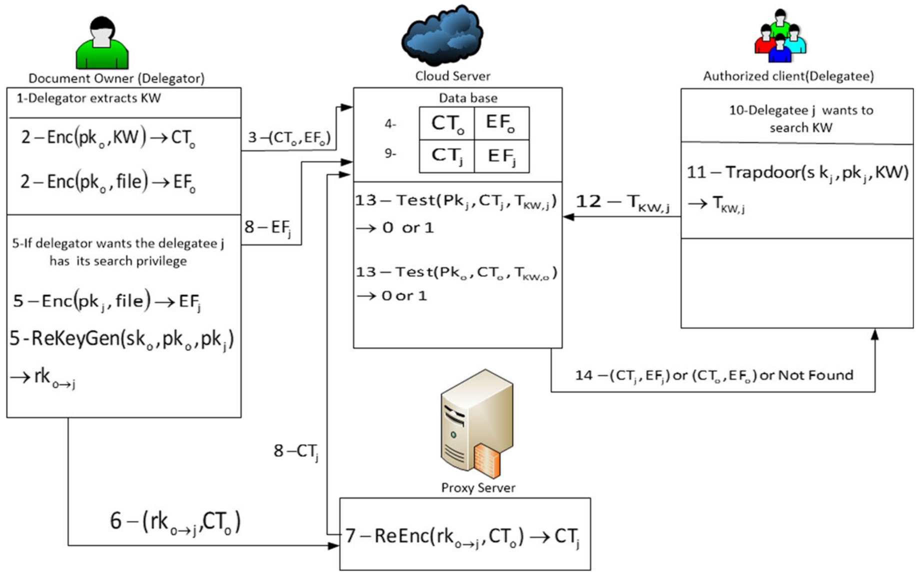

Document Owner (Delegator)
1-Delegator extracts KW
2 – Enc(pk_o, KW) → CT_o
2 – Enc(pk_o, file) → EF_o
5-If delegator wants the delegatee j has its search privilege
5 – Enc(pk_j, file) → EF_j
5-ReKeyGen(sk_o, pk_o, pk_j) → rk_o→j
Cloud Server
Data base
4- CT_o EF_o
9- CT_j EF_j
13 – Test(Pk_j, CT_j, T_KW,j) → 0 or 1
13 – Test(Pk_o, CT_o, T_KW,o) → 0 or 1
Authorized client(Delegatee)
10-Delegatee j wants to search KW
11 – Trapdoor(sk_j, pk_j, KW) → T_KW,j
3 –(CT_o, EF_o)
8 – EF_j
12 – T_KW,j
14 –(CT_j, EF_j) or (CT_o, EF_o) or Not Found
8 – CT_j
6 – (rk_o→j, CT_o)
Proxy Server
7 – ReEnc(rk_o→j, CT_o) → CT_j

Fig. 1 The Yang et al.'s searchable encryption scheme

2 Yang et al.'s Searchable Encryption Scheme

In [42], Yang et al. proposed a new searchable encryption based on lattice which supports semantic keyword search and proxy re-encryption functions. Their searchable encryption scheme includes the following four entities:

- *Document Owner (delegator)* who wants to store his documents in an encrypted form on the cloud server such that, if necessary, an authorized party (delegatee) should be able to search encrypted documents with the proper keyword.
- *Cloud Server* responsible for storing encrypted documents and also responsible for searching in the encrypted data to find encrypted files that include the certain keyword.
- *Proxy Server* responsible for converting data owner secure index CT_o to delegatee secure index CT_j.
- *Authorized Client (delegatee)* who has the permission to receive the data owners encrypted files.

It should be noted that the details of scheme's functions computations do not have any effect on the proposed attack and it is ignored in this paper, the interested reader can refer [42] for more details. Throughout the paper, we use the notations represented in Table 1. As shown in Fig. 1, the Yang et al.'s searchable encryption scheme works as follows:

1. *New user registration* in this phase, the trusted third party TTP considers a new user's identity and if it is valid, TTP generates public key pk_i and private key sk_i for the user by using $KeyGen(k) \rightarrow (pk_i, sk_i)$.
2. *Encrypted documents generation* in this phase, the document owner, before outsourcing the file to the cloud server, extracts a keyword KW from the file and generates a secure index $CT_o = Enc(pk_o, KW)$ and sends the encrypted file $EF_o = Enc(pk_o, file)$ and CT_o to the cloud server. In this phase, if the data owner (delegator) wants to allow the other (delegatee) to search, it has to generate an encrypted file with the public key of

delegatee (j), i.e. $EF_j = Enc(pk_j, file)$, and generate a re-encryption key. By sending that re-encryption key to the proxy server, the proxy server converts CT_o into CT_j, which is the secure index of delegatee (j).

3. *Re-encryption key generation* in this phase, if the user o wants to give its search privilege to user j, using $ReKeyGen(sk_o, pk_o, pk_j)$, it generates the re-encryption key $rk_{o \to j}$ which is sent along with CT_o to the proxy server, for secure index transformation. Whenever the proxy server receives $rk_{o \to j}$ and CT_o, it transforms CT_o to CT_j as $ReEnc(rk_{o \to j}, CT_o)$.
4. *Secure index transformation* in this phase, using $ReEnc(rk_{o \to j}, CT_o)$, the proxy server transforms CT_o of the user o (delegator) to the CT_j of user j (delegatee).
5. *Keyword trapdoor generation* if an authorized user j wants to find the encrypted documents including keyword KW, generates the trapdoor using his secret key sk_j as $Trapdoor(sk_j, pk_j, KW)$ and sends $T_{KW,j}$ to the server.
6. *Match files retrieving* the cloud server upon receipt the keyword trapdoor, searches the encrypted files that include the search keyword or its synonym through following relations:
   $Test(pk_o, CT_o = Enc(pk_o, KW), T_{KW,o} = Trapdoor(sk_o, pk_o, KW)) \overset{?}{=} 1$
   $Test(pk_j, CT_j = ReEnc(rk_{o \to j}, CT_o), T_{KW,j} = Trapdoor(sk_j, pk_j, KW)) \overset{?}{=} 1$
   If at least one of the above relations is satisfied, then the cloud server sends CT_x and EF_x where $x \in \{o, j\}$ to the user.

More details about Yang et al.'s searchable encryption scheme can be found in [42].

3 Security Analysis of Yang et al.'s Searchable Encryption Scheme

In this section, we show that how an adversary A can threaten the security of the Yang et al.'s searchable encryption scheme. Integrity contradiction attack is an attack for which an adversary tries to modify the encrypted files during its transfer without being realized by cloud server, authorized clients or data owner. If the adversary succeeds in changing the encrypted files and the cloud server saves the modified encrypted files, then the data owners or authorized clients will no longer be able to access the original encrypted data that was sent to the server. Given that most of the time, after data transfer, the data owner may clean it from its storage system, the suggested attack is very serious and it is necessary to counter searchable encryption schemes against it. On the other hand, changing the encrypted files during transfer to the data owners or authorized clients from the cloud server causes the data owner or client to receive the modified encrypted file, and this may result in irreparable damage due to incorrect information that reveals from the modified encrypted file.

To apply the integrity contradiction attack in the Yang *et al.*'s searchable encryption scheme, the adversary can do either of the following three scenarios as shown in Fig. 2:

Attack Scenario 1: This scenario runs as below:

1. A data owner:
   – extracts the keyword $KW \in \{0, 1\}^*$;
   – generates the secure index $CT_o = Enc(pk_o, KW)$;
   – generates the encrypted document $EF_o = Enc(pk_o, file)$;
   – and sends CT_o and EF_o to the cloud server.
2. The adversary:
   – stops CT_o and EF_o;

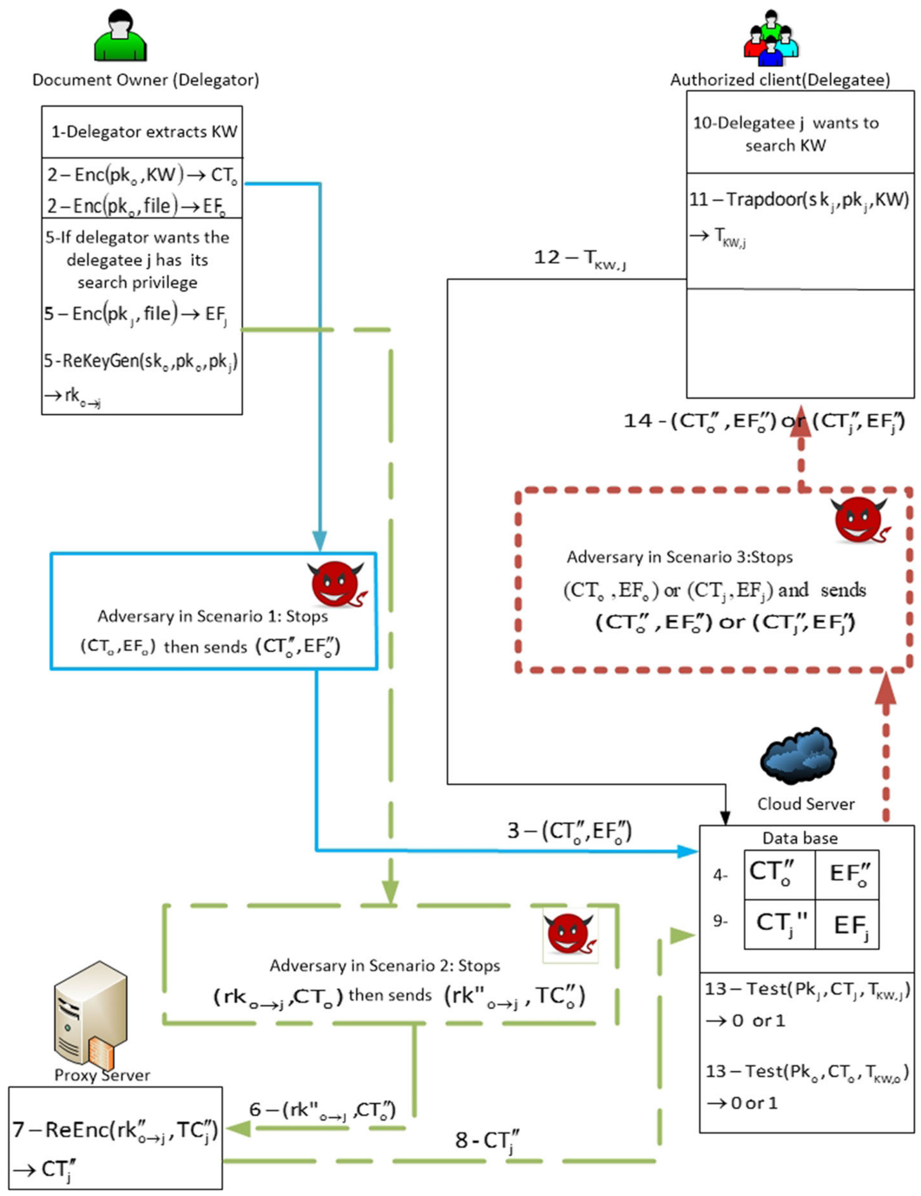

Document Owner (Delegator)
1-Delegator extracts KW
2 – Enc(pk_o, KW) → CT_o
2 – Enc(pk_o, file) → EF_o
5-If delegator wants the delegatee j has its search privilege
5 – Enc(pk_j, file) → EF_j
5-ReKeyGen(sk_o, pk_o, pk_j) → rk_o→j
Authorized client(Delegatee)
10-Delegatee j wants to search KW
11 – Trapdoor(sk_j, pk_j, KW) → T_KW,j
12 – T_KW,J
14 - (CT_o″, EF_o″) or (CT_j″, EF_j″)
Adversary in Scenario 3:Stops (CT_o, EF_o) or (CT_j, EF_j) and sends (CT_o″, EF_o″) or (CT_j″, EF_j″)
Adversary in Scenario 1: Stops (CT_o, EF_o) then sends (CT_o″, EF_o″)
Cloud Server
3 – (CT_o″, EF_o″)
Data base
4- CT_o″ EF_o″
9- CT_j″ EF_j
13 – Test(Pk_j, CT_j, T_KW,j) → 0 or 1
13 – Test(Pk_o, CT_o, T_KW,o) → 0 or 1
Adversary in Scenario 2: Stops (rk_o→j, CT_o) then sends (rk″_o→j, TC_o″)
Proxy Server
6 – (rk″_o→j, CT_o″)
7 – ReEnc(rk″_o→j, TC_o″) → CT_j″
8 - CT_j″

Fig. 2 The proposed integrity contradiction attack against the Yang et al.'s searchable encryption scheme

- changes CT_o to an arbitrary value, e.g. CT_o'', and also the encrypted file EF_o to an arbitrary file EF_o'' ;
- and sends CT_o'' and EF_o'' to the cloud server.

3. The cloud server receives CT_o'' and EF_o'' and stores them without any check which indicates that these files have changed during the transfer.
   Therefore, the modified secure index and modified encrypted file may not output correctly

in the keyword search phase, and thus the data owners or allowed clients by searching the keyword, along with the trapdoor, will not access the correct file. The success probability of this attack is one while its complexity is only one run of the scheme.

Attack Scenario 2: Given that the data owner (delegator) wants to allow the other (delegatee) to search, the second scenario to attack Yang et al.'s searchable encryption scheme runs as below:

1. The data owner:
   - generates the encrypted document $EF_j = Enc(pk_j, file)$;
   - sends EF_j to the cloud server;
   - generates a re-encryption key $rk_{o\to j}$ through $ReKeyGen(sk_o, pk_o, pk_j)$;
   - and sends CT_o and $rk_{o\to j}$ to the proxy server in the insecure channel.
2. The adversary:
   - stops CT_o and $rk_{o\to j}$;
   - changes CT_o and $rk_{o\to j}$ to an arbitrary value, e.g. CT_o'' and $rk''_{o\to j}$ respectively;
   - and sends CT_o'' and $rk''_{o\to j}$ to the proxy server.
3. The proxy server receives CT_o'' and $rk''_{o\to j}$ and without any checking transforms CT_o'' to CT_j'' using $rk''_{o\to j}$, which is not equal to the intended value of CT_j.
   So, the authorized user j after above attack, will never access the correct file. The success probability of above attack is one while its complexity is only one run of the scheme.

Attack Scenario 3: This scenario, given the encrypted files are correctly and without any change stored on the cloud server, runs as below:

1. The authorized client i.e. the user j requests cloud server for encrypted file which includes the KW via the trapdoor $T_{KW,j}$;
2. After the successful keyword search, the cloud server sends the secure index CT_j along with the encrypted document EF_j for it.
3. The adversary:
   - stops CT_j and EF_j;
   - changes CT_j to an arbitrary value CT_j'' and also EF_j to an arbitrary EF_j'' ;
   - and sends CT_j'' and EF_j'' to the delegatee (j).
4. Upon receipt CT_j'' and EF_j'', the authorized client j:
   - by using its secret key sk_j and through decryption of CT_j'' and EF_j'' finds a keyword and a file respectively which are different from original keyword and original file.

   So, according to the above attack scenario, allowed client will not access the correct file. The success probability of above attack is one while its complexity is only one run of the scheme.

Yang et al.'s searchable encryption scheme's vulnerability to above attack is due to the fact that the integrity of the messages exchanged in the protocol is not checked by the recipients of the messages.
It is worth noting that due to the fact that the communication channel is not secure, it is possible for everyone to eavesdrop. So, the attacker can receive the sent message and even change it and send it again. In fact, this attack is practically possible because there is no integrity check function in the transferred messages between protocol parties for example between the document owner and the cloud server, so that the cloud server does not realize

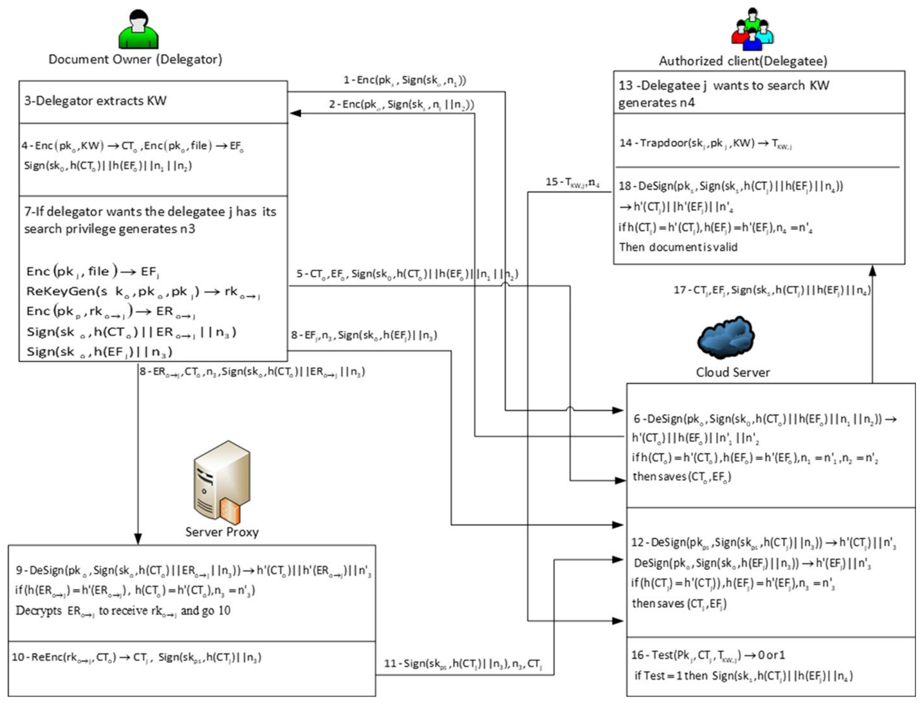
that the message has been changed. Hence, there is a possibility of transferred messages change or manipulation. For example, suppose the first message from the sender (owner of the file) is sent to the server and reaches the server in a matter of seconds without manipulation or interception, and the server saved it. Now suppose that in sending a message from the file owner to the server in one of the intermediate nodes, the attacker saves the message and then changes it (although he/she does not know the contents of the message due to encryption), for example, a few bits are converted from 0 to 1 or 1 to 0 and sends the changed message again. The message eventually reaches the cloud server, but it may take more the time to receive it. Due to the nature of computer networks and the momentary change in network traffic, the cloud server assumes this time increase is due to high network traffic and stores it as a healthy message on the cloud server. So this attack is practically feasible.

4 Improved Yang et al.'s Searchable Encryption

Given that the Yang et al.'s searchable encryption scheme does not guarantee the integrity of the stored data, in the previous section we proposed several efficient attacks against it. In this section, we improve this searchable encryption scheme which is resistant against attacks mentioned in Sect. 3 and the other known active and passive attacks. The attack described above is due to the weakness of the Yang et al.'s searchable encryption scheme, which in it neither the server nor the data owner nor the client checks the integrity of its received encrypted files. If such mechanism can be implemented, which in it before the encrypted file is stored on the cloud server and also upon the data owner or allowed clients receive the encrypted file and before using it, they make sure about the integrity preservation of the encrypted file, then the searchable encryption scheme resists against integrity contradiction attack. The basic idea of our improving is using digital signatures of the secure index and also digital signatures of encrypted documents which along with the secure index and the encrypted document are sent to all desired exchange paths of improved searchable encryption scheme. We used the RSA encryption algorithm, RSA digital signature algorithm, and sha256 hash algorithm in order to encrypt, sign and hash the messages respectively.

The improved Yang et al.'s searchable encryption, as it is shown in Fig. 3, runs as below:

- *New user registration* in this phase, TTP considers a new user's identity and if it is valid, the TTP generates public key pk_i and private key sk_i for the user by using $KeyGen(k) \rightarrow (pk_i, sk_i)$.
- *Encrypted documents generation* in this phase, the data owner, before outsourcing the file to the cloud server generates a random number n_1 and signs it with its secret key as $Sign(sk_o, n_1)$ and then encrypts it with public key of cloud server as $Enc(pk_s, Sign(sk_o, n_1))$ and sends it to the cloud server. Cloud server once receives the message decrypts the message and then receives n_1 by verifying the signature. If the cloud server is ready to accept files to outscoring, generates another random number n_2 and signs $n_1\|n_2$ with its secret key as $Sign(sk_o, n_1\|n_2)$ and then encrypts it with public key of delegator as $Enc(pk_o, Sign(sk_o, n_1\|n_2))$ and sends it to the delegator. Delegator as soon as receives the message, decrypts it and verifies signature and receives n_1 and n_2, if the received n_1 equals with sent n_1, then it:
    - extracts a keyword KW from the file;
    - generates a secure index CT_o as $Enc(pk_o, KW)$;
    - generates an encrypted document EF_o as $Enc(pk_o, file)$;



Document Owner (Delegator)
Authorized client(Delegatee)
3-Delegator extracts KW
4-Enc(pk_o,KW) → CT_o, Enc(pk_o,file) → EF_o Sign(sk_o,h(CT_o)||h(EF_o)||n_1||n_2)
7-If delegator wants the delegatee j has its search privilege generates n3
Enc(pk_j, file) → EF_j
ReKeyGen(sk_o,pk_o,pk_j) → rk_o→j
Enc(pk_p,rk_o→j) → ER_o→j
Sign(sk_o,h(CT_o)||ER_o→j||n_3)
Sign(sk_o,h(EF_j)||n_3)
1-Enc(pk_o, Sign(sk_j,n_1))
2-Enc(pk_o, Sign(sk_j,n_1||n_2))
5-CT_o,EF_o, Sign(sk_o,h(CT_o)||h(EF_o)||n_1||n_2)
8-EF_j,n_3, Sign(sk_o,h(EF_j)||n_3)
8-ER_o→j,CT_o,n_3,Sign(sk_o,h(CT_o)||ER_o→j||n_3)
13-Delegatee j wants to search KW generates n4
14-Trapdoor(sk_j,pk_j,KW) → T_KW,j
15-T_KW,j,n_4
18-DeSign(pk_s, Sign(sk_s,h(CT_j)||h(EF_j)||n_4)) → h'(CT_j)||h'(EF_j)||n'_4
if h(CT_j) = h'(CT_j), h(EF_j) = h'(EF_j), n_4 = n'_4
Then document is valid
17-CT_j,EF_j, Sign(sk_s,h(CT_j)||h(EF_j)||n_4)
Cloud Server
6-DeSign(pk_o, Sign(sk_o,h(CT_o)||h(EF_o)||n_1||n_2)) → h'(CT_o)||h(EF_o)||n'_1||n'_2
if h(CT_o) = h'(CT_o), h(EF_o) = h'(EF_o), n_1 = n'_1, n_2 = n'_2
then saves (CT_o,EF_o)
Server Proxy
9-DeSign(pk_o, Sign(sk_o,h(CT_o)||ER_o→j||n_3)) → h'(CT_o)||h'(ER_o→j)||n'_3
if (h(ER_o→j) = h'(ER_o→j), h(CT_o) = h'(CT_o), n_3 = n'_3)
Decrypts ER_o→j to receive rk_o→j and go 10
10-ReEnc(rk_o→j,CT_o) → CT_j, Sign(sk_ps,h(CT_j)||n_3)
11-Sign(sk_ps,h(CT_j)||n_3), n_3, CT_j
12-DeSign(pk_ps,Sign(sk_ps,h(CT_j)||n_3)) → h'(CT_j)||n'_3
DeSign(pk_o,Sign(sk_o,h(EF_j)||n_3)) → h'(EF_j)||n'_3
if (h(CT_j) = h'(CT_j)), h(EF_j) = h'(EF_j), n_3 = n'_3
then saves (CT_j,EF_j)
16-Test(Pk_j,CT_j,T_KW,j) → 0 or 1
if Test = 1 then Sign(sk_s,h(CT_j)||h(EF_j)||n_4)

Fig. 3 The proposed searchable encryption scheme

  - signs $h(CT_o)\|h(EF_o)\|n_1\|n_2$ by using its secret key sk_o as $Sign(sk_o, h(CT_o)\| h(EF_o)\|n_1\|n_2)$;
  - and sends CT_o, EF_o and $Sign(sk_o, h(CT_o)\|h(EF_o)\|n_1\|n_2)$ to the cloud server.

- *Encrypted documents integrity check* in this phase, upon receipt CT_o, EF_o and $Sign(sk_o, h(CT_o)\|h(EF_o)\|n_1\|n_2)$, the cloud server:
  - verifies the signature $Sign(sk_o, h(CT_o)\|h(EF_o)\|n_1\|n_2)$ by using the public key of owner pk_o through $DeSign(pk_o, Sign(sk_o, h(CT_o)\|h(EF_o)\|n_1\|n_2))$ and receives $h'(CT_o)\|h'(EF_o)\|n'_1\|n'_2$;
  - checks whether $h'(CT_o) \overset{?}{=} h(CT_o)$, $h'(EF_o) \overset{?}{=} h(EF_o)$, $n'_1 \overset{?}{=} n_1$ and $n'_2 \overset{?}{=} n_2$. If one of them does not hold, it does not store CT_o and sends an “Error” message to the data owner. If it is ok, stores CT_o and EF_o.

- *Re-encryption key generation* at this phase, if the data owner (delegator) wants to allow the other (delegatee) j to search, it has to:
  - generate a random number n_3;
  - generate an encrypted document EF_j as $Enc(pk_j, file)$;
  - sign $h(EF_j)\|n_3$ by using its secret key sk_o as $Sign(sk_o, h(EF_j)\|n_3)$;
  - send EF_j, n_3 along with $Sign(sk_o, h(EF_j)\|n_3)$ to the cloud server;
  - generate a re-encryption key $rk_{o\to j}$ as $ReKeyGen(sk_o, pk_o, pk_j)$;
  - generate an encryption of re-encryption key $rk_{o\to j}$ with the public key of proxy server as $ER_{o\to j} = Enc(pk_p, rk_{o\to j})$;

- sign $h(ER_{o\to j})\|h(CT_o)\|n_3$ as $Sign(sk_o, h(ER_{o\to j})\|h(CT_o)\|n_3)$;
- and send $ER_{o\to j}$, CT_o, n_3 and $Sign(sk_o, h(ER_{o\to j})\|h(CT_o)\|n_3)$ to the proxy server.

– *Secure index transformation* the proxy server when receives the messages:

- verifies $Sign(sk_o, h(ER_{o\to j})\|h(CT_o)\|n_3)$ with the public key of document owner pk_o through $DeSign(pk_o, Sign(sk_o, h(ER_{o\to j})\|h(CT_o)\|n_3))$ and receives $h'(ER_{o\to j})\|h'(CT_o)\|n'_3$;
- checks whether $h(ER_{o\to j}) \overset{?}{=} h'(ER_{o\to j})$, $h(CT_o) \overset{?}{=} h'(CT_o)$ and $n_3 \overset{?}{=} n'_3$. If one of them does not hold, it does not decrypt $ER_{o\to j}$ and sends an "Error" message to the data owner, otherwise:
  - ∗ decrypts $ER_{o\to j}$ with its secret key as $Dec(sk_p, ER_{o\to j}) = Dec(sk_p, Enc(pk_p, rk_{o\to j}))$ and obtains $rk_{o\to j}$;
  - ∗ by using the re-encryption key $rk_{o\to j}$ transforms CT_o to CT_j.
  - ∗ signs $h(CT_j)\|n_3$ as $Sign(sk_p, h(CT_j)\|n_3)$;
  - ∗ and sends CT_j, n_3 and $Sign(sk_p, h(CT_j)\|n_3)$ to the cloud server.

– *Keyword trapdoor generation* if an authorized user j wants to find the encrypted documents including keyword KW, produces a random number n_4, generates the trapdoor $T_{KW,j}$ using his secret key through $Trapdoor(sk_j, pk_j, KW)$ and sends $T_{KW,j}$ along with n_4 to the cloud server.

– *Match files retrieving* upon receipt the keyword trapdoor, the cloud server:

- searches the encrypted files that include the search keyword or its synonym through following relations:
- if $Test(pk_x, CT_x, T_{KW,x}) \to 0$, where $x \in \{o, j\}$ responds "cannot find the keyword in database";
- if $Test(pk_x, CT_x, T_{KW,x}) \to 1$, where $x \in \{o, j\}$ finds related CT_x and EF_x and signs $h(CT_x)\|h(EF_x)\|n_4$ with its secret key as $Sign(sk_s, h(CT_x)\|h(EF_x)\|n_4)$ and sends CT_x, EF_x and $Sign(sk_s, h(CT_x)\|h(EF_x)\|n_4)$ to the client.

– *Check the integrity of received encrypted files* the authorized client, when receives CT_j, EF_j and $Sign(sk_s, h(CT_j)\|h(EF_j)\|n_4)$:

- verifies the signature $Sign(sk_s, h(CT_j)\|h(EF_j)\|n_4)$ by using the public key of the cloud server pk_s as $DeSign(pk_s, Sign(sk_s, h(CT_j)\|h(EF_j)\|n_4))$ and receives $h'(CT_j)\|h'(EF_j)\|n'_4$;
- checks whether $h'(CT_j) \overset{?}{=} h(CT_j)$, $h'(EF_j) \overset{?}{=} h(EF_j)$ and $n'_4 \overset{?}{=} n_4$.
  - ∗ If one of them does not hold, does not use the encrypted files and sends another request to the cloud server;
  - ∗ If it is ok, stores encrypted file EF_j and decrypts it by using its secret key and consequently uses it.

Remark 1 In the proposed scheme, hash of messages is signed instead of signing the message itself, in order to increase the speed of sign operation.

5 Security Analysis of the Improved Searchable Encryption Scheme

In this section, we first informally and then formally show that the improved searchable encryption scheme can provide complete security against different attacks through the Scyther tool [11].

5.1 Informal Analysis

Since the improved searchable encryption scheme is based on the Yang et al.'s scheme, its security against other attacks is the same as the Yang et al.'s scheme security analysis which is presented in [42] and for the sake of brevity we do not explain them here. It is enough to show that the improved searchable encryption scheme resists against the attack represented in this paper.

5.1.1 Resistance to Integrity Contradiction Attack

In the improved searchable encryption scheme, wherever a message was transported, along with that, the signed message, would also be transmitted. Therefore, the recipient of the message could also ensure that the sender of the message is the one claiming and the message during the transfer, has not changed, which is interpreted as data integrity preservation. Hence, the improved searchable encryption scheme can resist against the integrity contradiction attack presented in this paper.

5.1.2 Resistance to Replay Attack

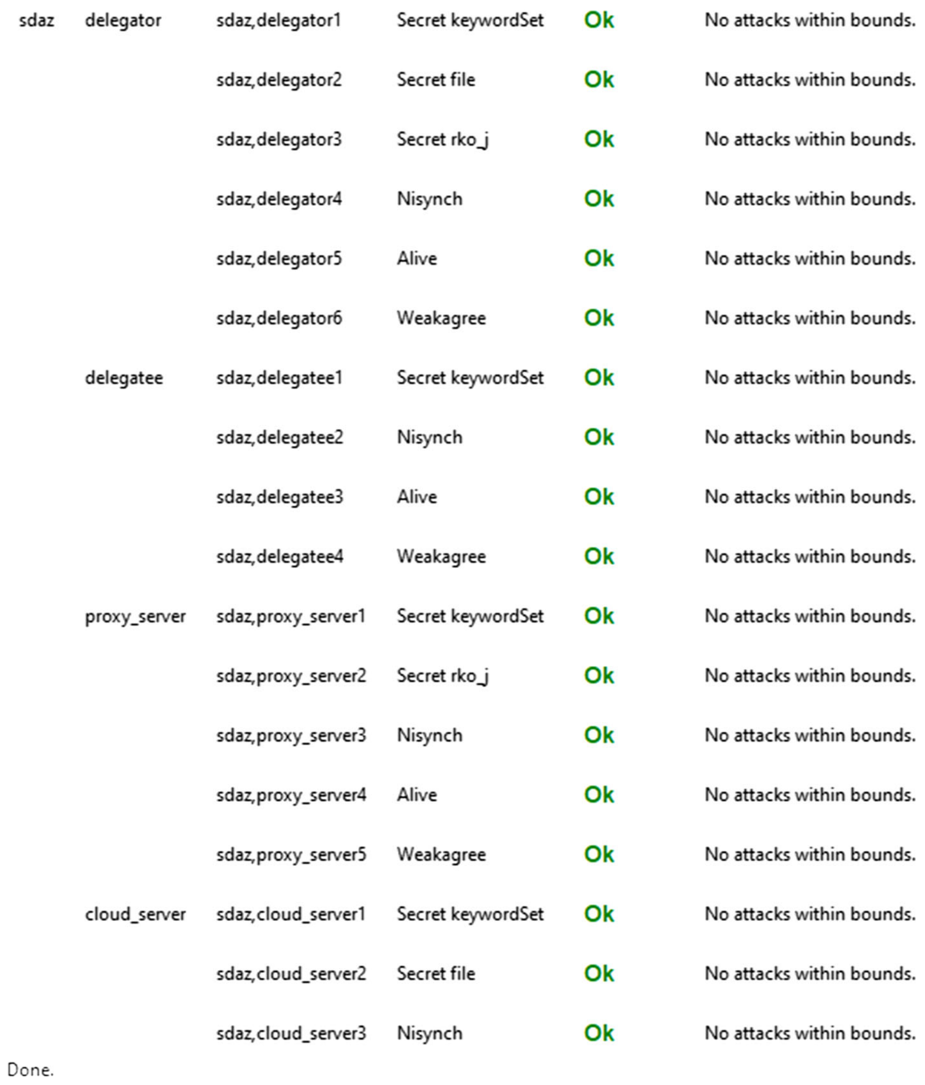
To resist against replay attacks in the proposed scheme, the protocol parties eliminate the possibility of reusing the messages by generating random numbers and using them in signed messages. Therefore, the proposed protocol is safe against a variety of replay attacks.

5.1.3 Resistance to Impersonation Attack

Since in the proposed scheme, each response to a request is signed with the respondent's private key and also, inside the signed messages, there is a random number sent by the initiator of the communication, so it is not possible for the adversary to impersonate a legitimate party. Therefore, the resistance of the proposed scheme to the all kinds of impersonation attacks is due to the signing of messages that have been randomized by the parties by generating random numbers.

5.2 Formal Analysis Based on Scyther Tool

Today, there are various tools for formal security analysis such as TAMARIN [28], AVISPA [2], Proverif [5], CryptoVerif [4] and Scyther [11]. Here, we consider the security of our proposed scheme in term of security using the Scyther tool. The Scyther is a tool to check the security properties of a protocol based on security claims. The Scyther tool takes the description of the protocol in security protocol description language (SPDL) structure as an input, and can define security claims for each role in the protocol, and check the correctness of claims output. If the claim is not correct, its attack structure is shown in graphical form.



sdaz	delegator	sdaz,delegator1	Secret keywordSet	Ok	No attacks within bounds.
		sdaz,delegator2	Secret file	Ok	No attacks within bounds.
		sdaz,delegator3	Secret rko_j	Ok	No attacks within bounds.
		sdaz,delegator4	Nisynch	Ok	No attacks within bounds.
		sdaz,delegator5	Alive	Ok	No attacks within bounds.
		sdaz,delegator6	Weakagree	Ok	No attacks within bounds.
	delegatee	sdaz,delegatee1	Secret keywordSet	Ok	No attacks within bounds.
		sdaz,delegatee2	Nisynch	Ok	No attacks within bounds.
		sdaz,delegatee3	Alive	Ok	No attacks within bounds.
		sdaz,delegatee4	Weakagree	Ok	No attacks within bounds.
	proxy_server	sdaz,proxy_server1	Secret keywordSet	Ok	No attacks within bounds.
		sdaz,proxy_server2	Secret rko_j	Ok	No attacks within bounds.
		sdaz,proxy_server3	Nisynch	Ok	No attacks within bounds.
		sdaz,proxy_server4	Alive	Ok	No attacks within bounds.
		sdaz,proxy_server5	Weakagree	Ok	No attacks within bounds.
	cloud_server	sdaz,cloud_server1	Secret keywordSet	Ok	No attacks within bounds.
		sdaz,cloud_server2	Secret file	Ok	No attacks within bounds.
		sdaz,cloud_server3	Nisynch	Ok	No attacks within bounds.

Done.

Fig. 4 The security verification results of the improved searchable encryption scheme in the Scyther tool

We implemented the proposed scheme in SPDL which its code is presented in Appendix A. The results of the automatic security verification of the proposed scheme by the Scyther tool as shown in Fig. 4 are verified OK.

6 Security Comparison

In this section, we will briefly compare the security features of our proposed scheme with schemes [27,37,39,41,42] in Table 2. The notations used in Table 2 are as follows:

Table 2 Security properties comparison

Schemes	KB	Resistance to KGA	Resistance to FIA	NI	Resistance to ICA
[27]	✓	✗	–	✗	–
[37]	✓	✓	✓	–	–
[39]	✓	✓	–	✓	–
[41]	✓	✓	–	✓	–
[42]	✓	–	–	✓	✗
Our scheme	✓	✓	✓	✓	✓

- *KB (Keyword-based)* The designed structures based on keywords are important and their functional capabilities are easy to find all relevant documents from the storing server by using the keyword search.
- *KGA (Keyword Guessing Attack)* In this attack, the external or internal adversary intends to find the exact amount of keywords by collecting information and eavesdropping on the insecure channels in the searchable encryption schemes.
- *FIA (File Injection Attack)* This attack was presented by Zhang et al. [45] aiming to discover the keywords associated with the files. In this attack, the attacker injects a certain number of files, each file containing at least the half of the total number of keywords, to the server. Then it tries to guess the keywords by getting the trapdoors from the data users, accessing the Test algorithms, and generating the ciphertext. More precisely, the attacker runs the attack as the following. Assume we have the keyword set KW, each of which is numbered with $log|KW|$ bits from 0 to $|KW| - 1$ (where $|KW|$ is the power of 2). The attacker generates a set of $\lceil log|KW| \rceil$ files F for injecting to the server. Each file has $|KW| - 1$ bits and contains exactly the half of the keyword set KW. The file i contains the keywords whose the ith most-significant bit is equal to 1. We consider a search results value $R = \{F_1, F_2, \ldots\}_2$, which is a binary number for the returned results from the storage server side. This attack is also called a binary-search attack.
- *NI (Non Interactivity)* It refers to the fact that two entities of the data user and the data owner do not need to interact with each other and interact with the entities involved in the protocol to generate their own keys.
- *ICA (Integrity Contradiction Attack)* In the improved searchable encryption scheme, wherever a message was transported, the signed message would also be transmitted along with that. Therefore, the recipient of the message could also ensure that the sender of the message is the one claiming and the message has not changed during the transfer which is interpreted as data integrity preservation. Hence, the improved searchable encryption scheme can resist against integrity contradiction attack.

7 Comparisons

There are many algorithms for doing digital signature, and it is not necessary to detail the implementation in most cases unless a change is made in the process of digital signature algorithm, which leads to propose of a new digital signature algorithm. In our proposed searchable encryption scheme, it does not matter which digital signature algorithm is used. So, we implemented our proposed searchable encryption scheme using RSA encryption, RSA digital signature and sha256 hash algorithms. Implementation results of our proposed

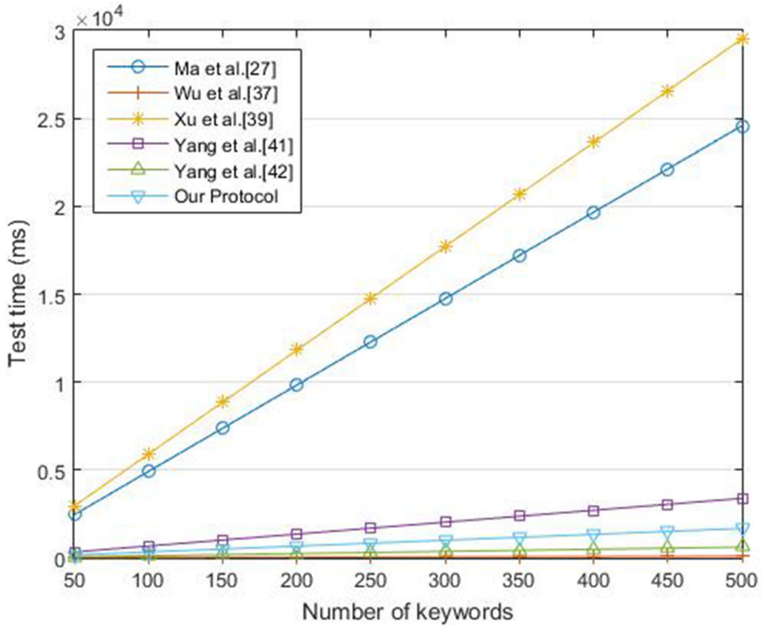


Fig. 5 The time required for Encryption algorithm in our proposed searchable encryption scheme compared to other related schemes

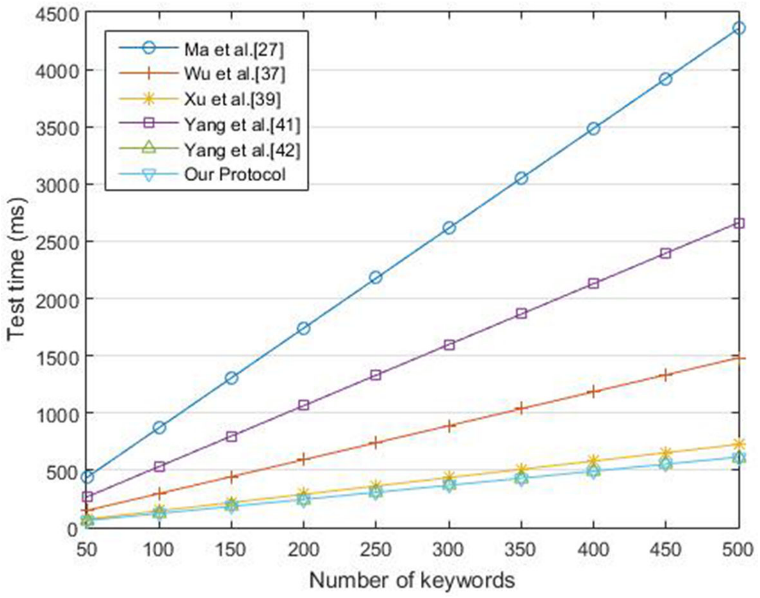


Fig. 6 The time required for Trapdoor algorithm in our proposed searchable encryption scheme compared to other related schemes

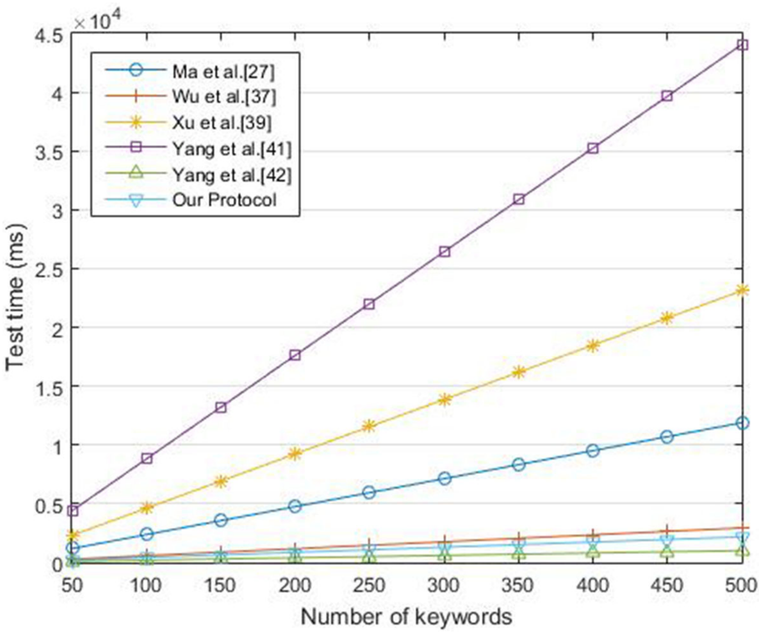


Fig. 7 The time required for Test algorithm in our proposed searchable encryption scheme compared to other related schemes

searchable encryption scheme compared to other related schemes are shown graphically in Figs. 5, 6, 7 and numerically in Tables 3, 4 and 5.

The system used for implementation has Intel(R) Core(TM) i5-4210u CPU@1.70GHz 2.40GHz CPU, 6 GB RAM and 1TB Hard. Table 3 shows the computational times of the proposed scheme compared to other schemes. In this table, the value of m (keyword set size) and l (encrypted keyword set size) are considered equal to 5 and 10 for [37,39] and [41] schemes, respectively. Table 4 shows the types of times used in a variety of searchable encryption schemes where T_p, T_{mtp}, T_{sm}, T_{exp}, T_{mul}, T_e, T_d, and T_h denote the required time for doing bilinear paring, doing map to point hash function, scalar multiplication operation, the exponentiation operation, the multiplication operation, the encryption operation, the decryption operation, and the typical hash operation respectively. Table 5 compares the proposed scheme with the related schemes in terms of communication costs. In this table, the communication cost is calculated based on the number of keywords sent in the communication channel. The graphical representation of our proposed scheme compared to other related schemes are shown in Figs. 5, 6 and 7 respectively.

Figure 5 shows that the Wu et al.'s scheme [37], the Yang et al.'s scheme [42] and the proposed scheme, require the least amount of time for their encryption algorithm, respectively. Figure 6 shows that the time required to generate a trapdoor is the same in both the Yang et al.'s scheme [42] and the proposed scheme, which is the lowest possible value compared to other designs. Figure 7 also shows that the proposed scheme and the Yang et al.'s scheme [42] require the least possible time for the test algorithm, respectively. In general, it can be said that although the total time of the improved searchable encryption scheme is slightly longer than its predecessor, but in contrast to it, it has been able to provide more security.

Table 3 Computational cost comparison

Schemes	Encryption phase (msec)	Trapdoor phase (msec)	Test phase (msec)
[27]	$3T_{mtp} + T_h + 4T_{sm} + 3T_p + T_{mul} = 49.154$	$T_{mtp} + T_{sm} + T_{mul} = 8.713$	$2T_{mtp} + T_h + T_{sm} + 2T_{mul} + T_p = 23.787$
[37]	$T_e + 2T_h = 0.199$	$3T_{sm} + T_h = 2.963$	$T_d = 5.896$
[39]	$(l^2 + 2l + 2)T_{exp} = 59.048$	$3T_{exp} = 1.452$	$l.m.T_{exp} + 3T_p = 46.226$
[41]	$(l + 4)T_{exp} = 6.776$	$(l + 1)T_{exp} = 5.324$	$(l + 2)T_p = 88.104$
[42]	$T_{ENC-RSA} = 1.230$	$T_{ENC-RSA} = 1.230$	$T_{DEC-RSA} = 2.040$
Our scheme	$T_{ENC-RSA} + T_{SIG-NRSA} = 3.350$	$T_{ENC-RSA} = 1.230$	$T_{DEC-RSA} + T_{SIGN-RSA} + 2T_h = 4.370$

Table 4 Times of operations used in searchable encryption schemes

Operation (ms)	T_p	T_{mtp}	T_{sm}	T_{exp}	T_{mul}
Time	7.342	7.726	0.986	0.484	0.001
T_e	T_d	T_h	$T_{ENC-RSA}$	$T_{DEC-RSA}$	$T_{SIGN-RSA}$
0.189	5.896	0.005	1.230	2.040	2.320

Table 5 Communication cost comparison where m and l for the schemes of [37,39,41], show the size of the keyword set and the size of the encrypted keyword set respectively

Schemes	Encryption phase	Trapdoor phase	Test phase
[27]	1	1	0
[37]	m	1	0
[39]	$l+2$	$m+3$	0
[41]	$l+3$	$l+3$	0
[42]	1	1	1
Our scheme	2	1	2

8 Conclusions

Searchable encryption schemes should have the capability to maintain an exchanged message's integrity, like any security algorithm or protocol. It means the message recipient can detect if the messages were changed during the transfer. In this paper, we have shown that the Yang et al.'s searchable encryption scheme does not provide the integrity of encrypted files during transfer over an insecure channel between scheme parties. More precisely, we have presented an integrity contradiction attack against this protocol with the success probability of one and the complexity of only one run of the scheme. Then we have proposed a new scheme by using digital signatures and have shown that the improved scheme resists to the attack presented in this paper and the other known active and passive attacks. This paper and the similar researches show that more work is needed in terms of the security of searchable encryption schemes.

A SPDL Implementation of the Proposed Scheme

```
usertype String;
usertype key;
hashfunction h;
const concat :Function;
protocol sdaz (delegator,cloud-server,proxy-server,delegatee)
{ role delegator{
secret file: String;
secret keywordSet: String;
fresh n1;
var n2;
fresh n3;
```

```
send_1(delegator,cloud-server,{{n1}sk(delegator)}
 pk(cloud-server));
recv_2(cloud-server,delegator,{{concat(n1,n2)}
 sk(cloud-server)}pk(delegator));
send_3(delegator,cloud-server,{keywordSet}pk(delegator),
 {file}pk(delegator),
 {concat(h({keywordSet}pk(delegator)),h({file}pk(delegator)),
 n1,n2)}sk(delegator));
send_4(delegator,cloud-server,{file}pk(delegatee),n3,
 {concat(h({file}pk(delegatee)),n3)}sk(delegator));
send_5(delegator,proxy-server,n3,{rko-j}pk(proxy-server),
{keywordSet}pk(delegator),
{concat(h({rko-j}pk(proxy-server)),h({keywordSet}
 pk(delegator)),n1,n2)}sk(delegator));
claim(delegator, Secret, keywordSet);
claim(delegator, Secret, file);
claim(delegator, Secret, rko-j);
claim(delegator, Nisynch);
claim(delegator, Alive);
claim(delegator, Weakagree);
    }
role delegatee{
secret file: String;
secret keywordSet: String;
fresh n4;
var n3;
send_7(delegatee,cloud-server,n4,{{keywordSet}pk(delegatee),
 n4}sk(delegatee));
recv_8(cloud-server,delegatee,{keywordSet}pk(delegatee),
 {file}pk(delegatee),
{concat(h({keywordSet}pk(delegatee)),h({file}pk(delegatee)),
 n4)}sk(cloud-server));
claim(delegatee, Secret, keywordSet);
claim(delegatee, Nisynch);
claim(delegatee, Alive);
claim(delegatee,Weakagree);
    }

role proxy-server{
secret rko-j: key;
secret keywordSet: String;
var n1;
var n2;
var n3;
recv_5(delegator,proxy-server,n3,{rko-j}pk(proxy-server),
 {keywordSet}pk(delegator),
{concat(h({rko-j}pk(proxy-server)),h({keywordSet}
 pk(delegator)),n1,n2)}sk(delegator));
send_6(proxy-server,cloud-server,{{keywordSet}pk(delegatee)}
 k(rko-j),n3,
{concat(h({keywordSet}pk(delegatee)),n3)}sk(proxy-server));
claim(proxy-server, Secret, keywordSet);
claim(proxy-server, Secret, rko-j);
claim(proxy-server, Nisynch);
claim(proxy-server, Alive);
claim(proxy-server, Weakagree);
    }
role cloud-server{
secret file: String;
secret keywordSet: String;
secret rko-j: key;
fresh n2;
var n1;
var n3;
var n4;
recv_1(delegator,cloud-server,{{n1}sk(delegator)}
 pk(cloud-server));
send_2(cloud-server,delegator,{{concat(n1,n2)}
 sk(cloud-server)}pk(delegator));
 recv_3(delegator,cloud-server,{keywordSet}pk(delegator),
 {file}pk(delegator),
   {concat(h({keywordSet}pk(delegator)),h({file}pk(delegator)),
 n1,n2)}sk(delegator));
recv_4(delegator,cloud-server,{file}pk(delegatee),n3,
 {concat(h({file}pk(delegatee)),n3)}sk(delegator));
 recv_6(proxy-server,cloud-server,{{keywordSet}pk(delegatee)}
 k(rko-j),n3,
 {concat(h({keywordSet}pk(delegatee)),n3)}sk(proxy-server));
recv_7(delegatee,cloud-server,n4,{{keywordSet}pk(delegatee),
 n4}sk(delegatee));
send_8(cloud-server,delegatee,{keywordSet}pk(delegatee),
 {file}pk(delegatee),
{concat(h({keywordSet}pk(delegatee)),h({file}pk(delegatee)),
 n4)}sk(cloud-server));
claim(cloud-server, Secret, keywordSet);
claim(cloud-server, Secret, file);
claim(cloud-server, Nisynch);
claim(cloud-server, Alive);
claim(cloud-server, Weakagree);
    }
}
```

References

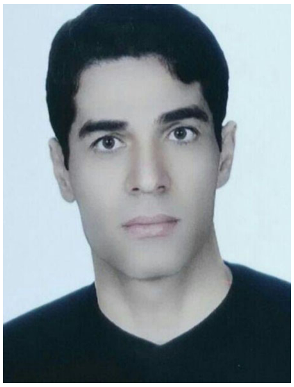
1. Abdelraheem, M. A., Andersson, T., & Gehrmann, C. (2017). Inference and record-injection attacks on searchable encrypted relational databases. *IACR Cryptology ePrint Archive*, *2017*, 24.
2. Armando, A., Basin, D., Boichut, Y., Chevalier, Y., Compagna, L., Cuéllar, J., Drielsma, P. H., Héam, P.-C., Kouchnarenko, O., & Mantovani, J., et al. (2005). The AVISPA tool for the automated validation of internet security protocols and applications. In *International conference on computer aided verification* (pp. 281–285). Springer.
3. Ateniese, G., Fu, K., Green, M., & Hohenberger, S. (2006). Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Transactions on Information and System Security*, *9*(1), 1–30.
4. Blanchet, B. (2007). CryptoVerif: Computationally sound mechanized prover for cryptographic protocols. In *Dagstuhl seminar "Formal Protocol Verification Applied"* (Vol. 117, p. 156).
5. Blanchet, B. (2013). Automatic verification of security protocols in the symbolic model: The verifier proverif. In *Foundations of Security Analysis and Design VII* (pp. 54–87). Springer.
6. Blaze, M., Bleumer, G., & Strauss, M. (1998). Divertible protocols and atomic proxy cryptography. In *International conference on the theory and applications of cryptographic techniques* (pp. 127–144). Springer.
7. Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004). Public key encryption with keyword search. In *International conference on the theory and applications of cryptographic techniques* (pp. 506–522). Springer.
8. Canetti, R., & Hohenberger, S. (2007). Chosen-ciphertext secure proxy re-encryption. In *Proceedings of the 14th ACM conference on Computer and communications security* (pp. 185–194). ACM.
9. Chang, Y.-C., & Mitzenmacher, M. (2005). Privacy preserving keyword searches on remote encrypted data. In *International conference on applied cryptography and network security* (pp. 442–455). Springer.
10. Chen, B., Wu, L., Li, L., Choo, K.-K. R., & He, D. (2020). A parallel and forward private searchable public-key encryption for cloud-based data sharing. *IEEE Access*, *8*, 28009–28020.
11. Cremers, C. J. F. (2008). The Scyther tool: Verification, falsification, and analysis of security protocols. In A. Gupta & S. Malik (Eds.), *Computer Aided Verification* (pp. 414–418). Berlin: Springer.
12. Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2011). Searchable symmetric encryption: Improved definitions and efficient constructions. *Journal of Computer Security*, *19*(5), 895–934.
13. Deng, Z., Li, K., Li, K., & Zhou, J. (2017). A multi-user searchable encryption scheme with keyword authorization in a cloud storage. *Future Generation Computer Systems*, *72*, 208–218.
14. Elizabeth, B. L., & Prakash, A. J. (2020). Verifiable top-k searchable encryption for cloud data. *Sādhanā*, *45*(1), 1–16.
15. Giraud, M., Anzala-Yamajako, A., Bernard, O., & Lafourcade, P. (2017). Practical passive leakage-abuse attacks against symmetric searchable encryption. *IACR Cryptology ePrint Archive*, *2017*, 46.
16. Goh, E.-J., et al. (2003). Secure indexes. *IACR Cryptology ePrint Archive*, *2003*, 216.
17. Grubbs, P., Ristenpart, T., & Shmatikov, V. (2017). Why your encrypted database is not secure. In *Proceedings of the 16th workshop on hot topics in operating systems* (pp. 162–168). ACM.
18. Grubbs, P., Sekniqi, K., Bindschaedler, V., Naveed, M., & Ristenpart, T. (2017). Leakage-abuse attacks against order-revealing encryption. In *2017 IEEE symposium on security and privacy (SP)* (pp. 655–672). IEEE.
19. Guo, C., Chen, X., Jie, Y., Zhangjie, F., Li, M., & Feng, B. (2017). Dynamic multi-phrase ranked search over encrypted data with symmetric searchable encryption. *IEEE Transactions on Services Computing*, *PP*(99), 1.
20. Huang, K., Tso, R., & Chen, Y.-C. (2017). Somewhat semantic secure public key encryption with filtered-equality-test in the standard model and its extension to searchable encryption. *Journal of Computer and System Sciences*, *89*, 400–409.
21. Jiang, P., Mu, Y., Guo, F., & Wen, Q.-Y. (2017). Private keyword-search for database systems against insider attacks. *Journal of Computer Science and Technology*, *32*(3), 599–617.
22. Jiang, X., Ge, X., Yu, J., Kong, F., Cheng, X., & Hao, R. (2017). An efficient symmetric searchable encryption scheme for cloud storage. *Journal of Internet Services and Information Security*, *2*, 1–18.
23. Li, C.-T., Lee, C.-C., Weng, C.-Y., Wu, T.-Y., & Chen, C.-M. (2017). Cryptanalysis of "an efficient searchable encryption against keyword guessing attacks for shareable electronic medical records in cloud-based system". In *International conference on information science and applications* (pp. 282–289). Springer.
24. Li, J., Qin, C., Lee, P. P., & Zhang, X. (2017). Information leakage in encrypted deduplication via frequency analysis. In *2017 47th Annual IEEE/IFIP international conference on dependable systems and networks (DSN)* (pp. 1–12). IEEE.

25. Liu, G., Yang, G., Bai, S., Zhou, Q., & Dai, H. (2020). FSSE: An effective fuzzy semantic searchable encryption scheme over encrypted cloud data. *IEEE Access*, *8*, 71893–71906.
26. Liu, Z., Li, T., Li, P., Jia, C., & Li, J. (2018). Verifiable searchable encryption with aggregate keys for data sharing system. *Future Generation Computer Systems*, *78*, 778–788.
27. Ma, M., He, D., Kumar, N., Choo, K.-K. R., & Chen, J. (2018). Certificateless searchable public key encryption scheme for industrial Internet of Things. *IEEE Transactions on Industrial Informatics*, *14*(2), 759–767.
28. Meier, S., Schmidt, B., Cremers, C., & Basin, D. (2013). The TAMARIN prover for the symbolic analysis of security protocols. In *International conference on computer aided verification* (pp. 696–701). Springer.
29. Miao, Y., Tong, Q., Deng, R., Choo, K.-K. R., Liu, X., & Li, H. (2020). Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage. *IEEE Transactions on Cloud Computing*,. https://doi.org/10.1109/TCC.2020.2989296.
30. Phuong, T. V. X., Yang, G., Susilo, W., Guo, F., & Huang, Q. (2017). Sequence aware functional encryption and its application in searchable encryption. *Journal of Information Security and Applications*, *35*, 106–118.
31. Poh, G. S., Chin, J.-J., Yau, W.-C., Choo, K.-K. R., & Mohamad, M. S. (2017). Searchable symmetric encryption: Designs and challenges. *ACM Computing Surveys (CSUR)*, *50*(3), 40.
32. Shao, J., & Cao, Z. (2009). CCA-secure proxy re-encryption without pairings. *International Workshop on Public Key Cryptography* (pp. 357–376). Berlin: Springer.
33. Shao, J., Cao, Z., Liang, X., & Lin, H. (2010). Proxy re-encryption with keyword search. *Information Sciences*, *180*(13), 2576–2587.
34. Song, D. X., Wagner, D., & Perrig, A. (2000). Practical techniques for searches on encrypted data. In *Proceedings of the 2000 IEEE symposium on security and privacy, 2000. S&P 2000* (pp. 44–55). IEEE.
35. Van Rompay, C., Molva, R., & Önen, M. (2017). A leakage-abuse attack against multi-user searchable encryption. *Proceedings on Privacy Enhancing Technologies*, *3*, 164–174.
36. Wang, X. A., Huang, X., Yang, X., Liu, L., & Wu, X. (2012). Further observation on proxy re-encryption with keyword search. *Journal of Systems and Software*, *85*(3), 643–654.
37. Wu, L., Chen, B., Choo, K.-K. R., & He, D. (2018). Efficient and secure searchable encryption protocol for cloud-based Internet of Things. *Journal of Parallel and Distributed Computing*, *111*, 152–161.
38. Wu, T.-Y., Meng, C., Chen, C.-M., Wang, K.-H., & Pan, J.-S. (2017). On the security of a certificateless public key encryption with keyword search. In *International conference on intelligent information hiding and multimedia signal processing* (pp. 191–197). Springer.
39. Xu, L., Li, J., Chen, X., Li, W., Tang, S., & Wu, H.-T. (2019). Tc-PEDCKS: Towards time controlled public key encryption with delegatable conjunctive keyword search for Internet of Things. *Journal of Network and Computer Applications*, *128*, 11–20.
40. Yang, Y., Liu, X., Deng, R. H., & Weng, J. (2017). Flexible wildcard searchable encryption system. *IEEE Transactions on Services Computing*, *13*, 464–477.
41. Yang, Y., & Ma, M. (2015). Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds. *IEEE*, *11*(4), 1.
42. Yang, Y., Zheng, X., Chang, V., & Tang, C. (2017). Semantic keyword searchable proxy re-encryption for postquantum secure cloud storage. *Concurrency and Computation: Practice and Experience*, *29*(19), e4211.
43. Ye, J., Wang, J., Zhao, J., Shen, J., & Li, K.-C. (2017). Fine-grained searchable encryption in multi-user setting. *Soft Computing*, *21*(20), 6201–6212.
44. Zhang, L., Su, J., & Mu, Y. (2020). Outsourcing attributed-based ranked searchable encryption with revocation for cloud storage. *IEEE Access*, *8*, 104344–104356.
45. Zhang, Y., Katz, J., & Papamanthou, C. (2016). All your queries are belong to us: The power of file-injection attacks on searchable encryption. In *25th* {USENIX} *Security Symposium* ({USENIX} *Security 16)* (pp. 707–720).
46. Zheng, Y., Lu, R., Shao, J., Yin, F., & Zhu, H. (2020). Achieving practical symmetric searchable encryption with search pattern privacy over cloud. *IEEE Transactions on Services Computing*,. https://doi.org/10.1109/TSC.2020.2992303.

Mohammad Zamani received his BS in Computer Engineering from Jundishapoor University of Dezful in 2016 and is currently an MScspsdot student in Computer Engineering at Shahid Rajaee Teacher Training Universityspsdot His research interests include Searchable Encryption in databases and Internet of Thingsspsdot

Masoumeh Safkhani is an assistant professor at the Computer Engineering Departmentspscomma Shahid Rajaee Teacher Training Universityspscomma Tehranspscomma Iranspsdot She received her PhspsdotDspsdot from Iran University of Science and Technologyspsdot She is the author of over 50 articles in information security and cryptologyspsdot

Negin Daneshpour is an assistant professor in the Computer Engineering faculty of Shahid Rajaee Teacher Training Universityspscomma Tehranspscomma Iranspsdot She received her BS degree in computer engineering hardware from the department of electronics and computer engineering at Shahid Beheshti Universityspscomma Iranspscomma where she graduated summa cum laude in 1999spsdot She received an MS degree and PhspsdotDspsdot in computer engineering-software from the Department of Computer Engineering and Information Technology at the Amirkabir University of Technologyspscomma Iranspscomma in 2002 and 2010spscomma respectivelyspsdot Her research interests focus on data analysis and managementspscomma data miningspscomma and data preprocessingspsdot



Amir Abbasian received the BS degree in software engineering from Shabestar Azad University and the MS degree in software engineering from Shahid Rajaee Teacher Training University in 2017spsdot His research areas are network securityspscomma RFID systems and wireless sensors networksspsdot