Volume XXIII 2020
ISSUE no.1
MBNA Publishing House Constanta 2020

Scientific Bulletin of Naval Academy

SBNA PAPER • OPEN ACCESS

Secure Document Search in Cloud Computing using MapReduce

To cite this article: Stefania Loredana Nita, Marius Iulian Mihailescu and Ciprian Racuciu, Scientific Bulletin of Naval Academy, Vol. XXIII 2020, pg.231-235.

ISSN: 2392-8956; ISSN-L: 1454-864X

doi: 10.21279/1454-864X-20-I1-031

Secure Document Search in Cloud Computing using MapReduce

Stefania Loredana Nita[1]; Marius Iulian Mihailescu[2]; Ciprian Racuciu[3]

[1]Computer Science Department, University of Bucharest;
[2]R&D Department, Dapyx Solution Ltd.;
[3]Computer Science Department, «Titu Maiorescu» University
stefanialoredanani@gmail.com; marius.mihailescu@hotmail.com;
ciprian.racuciu@gmail.com

Abstract. Nowadays, cloud computing is an important technology, which is part of our daily lives. Moving to cloud brings some benefits: create new applications, store large sets of data, process large amount of data. Individual users or companies can store own data on cloud (e.g. maritime, environmental protection, physics analysis etc.). An important thing before storing in cloud is that data needs to be encrypted, in order to keep its confidentiality. Among these, users can store encrypted documents on cloud. However, when owner needs a specific document, they should retrieve all documents from cloud, decrypt them, chose the desired document, encrypt again and finally store back encrypted documents on cloud. To avoid these entire steps, a user can choose to work with searchable encryption. This is an encryption technique, where key words (or indexes) are associated to encrypted documents, and when the owner needs a document, he/she only needs to search throw key words and then retrieve the documents that have associated the desired keywords. An important programming paradigm for cloud computing is MapReduce, which allows high scalability on a large number of servers in a cluster. Basically, MapReduce works with (key, value) pairs. In the current study paper, we describe a new technique through which a user can extract encrypted documents stored on cloud servers based on key words, using searchable encryption and MapReduce.

1. Introduction

Cloud computing represents an important technology nowadays, which has growth constantly in the IT environment. Cloud computing is so powerful, it can host or make working other technologies, such as Internet of Things (IoT), artificial intelligence (AI), or blockchain. There are more reasons for which cloud computing is attractive for both companies and individual users: scalability, reliability, elasticity, costs.

Still, due to its characteristics, cloud computing has some security issues, inherited from its components. To overcome these concerns, the research community proposed different techniques through which cloud computing can be securely used. One of these technique is searchable encryption (SE), which allow keyword searches over the encrypted data. For example, the staff from a medical center may search for a patient's data into a large amount of medical encrypted data stored in a cloud environment. In this way, the patient's data remain confidential, being read only by authorized staff.

Cloud computing has a great potential, which allows many complex computations. An important model that is widely used in cloud computing is MapReduce, which processes and generates large data sets in parallel and distributed over a cluster. The data in MapReduce model is stored into Hadoop

Distributed File System (HDFS). This is the reasons for which we use an MapReduce implementation along with a general searchable encryption scheme.

In the current study, we describe a general technique through which the search process of a searchable encryption scheme becomes faster, based on MapReduce model. Further, the paper is organized as follows: in the remaining section we present some concepts about searchable encryption and MapReduce, in section 2 we present the current state of the art for searchable encryption, the in the next section 3 we present the framework. Lastly, we will see some conclusions.

1.1. Searchable Encryption

Searchble encryption (SE) schemes are that schemes that allow search queries over encrypted data. There are two types of searchable encryption schemes: *symmetric searchable encryption* (SSE) and *public-key encryption with keyword search* (PEKS). In SSE just one key is used for both encryption and decryption and for the algorithms that require a key, namely a secret key, while in PEKS two keys are used, a public key for encryption and a secret key for decryption.

The entities that are involved in a SE scheme are: the *data owner* (this user owns the data, prepares and encrypts the data and send it to the server, establishes who may submit trapdoors to server), the *data user* (this user may submit trapdoor to the server and eventually to decrypt the result) and the *server* (stores the data in encrypted format and performs the search process).

Further, we present the algorithms in PEKS. The public-key searchable encryption schemes contain the following time polynomial algorithms:

- $KeyGen(\lambda) \rightarrow (k_p, k_s)$: The input for this algorithm is a security parameter λ and the output is a pair of keys (k_p, k_s). These keys are needed in different algorithms in the PEKS scheme.
- $Enc(k_p, D) \rightarrow C$: This is the encryption algorithm, whose input is: the public key k_p and a set of plan documents $Doc = \{ d_1, \dots, d_n \}$. The output is the encrypted list $= \{c_1, \dots, c_n \}$, where c_i repents the document $d_i, i = 1 \dots n$ in encrypted format.
- $BuildInd(D_i, w, K_{pub}) \rightarrow I$: Data owner runs this algorithm in order to create an index structure, which will keep the indexes of the documents. The algorithm takes as input the document D_i, the keyword w that describes the document and the public key K_{pub}. The result is an index structure *I* with the indexes associated to documents.
- $Trpd(k_s, w) \rightarrow t_w$: When the data user wants to search for a keyword, he/she run the trapdoor algorithm, whose result will be submitted to server. The input for trapdoor is the secret key k_s and the desired keyword w while the output is the value t_w.
- $Test\left(k_p, t_w, Enc(k_p, w')\right) \rightarrow b$: The input for the Test algorithm is the public key k_p, the trapdoor for w and the encryption of an word w'. Th output will be $b \in \{0, 1\}$, with $b = 1$ if $w' = w$ or $b = 0$, otherwise.
- $Dec(k_s, C) \rightarrow D$: Data user decrypts the encrypted document C, using the private key, resulting the document D in plain text.

1.2. MapReduce

MapReduce represents a programming model, designed predominantly for large amounts of data. It consist in algorithms that run in parallel, distributed on a cluster. This model is based on nodes that will perform the operations, which are classified in one *master* node and the rest of the nodes are called *slaves*.

The initial phase in *MapRreduce* is to scan the input from the user and to distribute it across the cluster. When the user uploads his/her files into the cloud, the initial phase will split the input into more *InputSplits*, of a predefined size s_{IS} (given by a parameter of system). If the total size of the input is s_T, then the number of slave nodes is equal to $\lceil \frac{s_T}{s_{IS}} \rceil$, because every slave will receive one block from split inputs.

Map. In the *map* phase the data is processed. In this phase, the following will happen: the slave nodes will transform into "mapper" nodes and every mapper will process its own input. In this phase, the mappers will initially apply *Scan* function on the input, resulting a list of pairs in the form *(key, value)*. Next, the mappers will apply the *Map* function, which will lead to an intermediary list of pairs *(key, value)*.

Reduce. The input for the *reduce* phase is the list of all intermediary pairs from all mappers. In this phase, the slave nodes will transform into "reducer" nodes. Every reducer will apply the *reduce* function, in which a reducer is selected for every pair with the same key. The output is written into a file and sent to user.

2. Related Work

In order to keep the security, but a fast search too, more encryption schemes with keyword search for cloud environment were proposed. The former SE scheme was proposed in [1], where the proposed scheme achieved controlled searching (only authorized users may submit trapdoor to the server), hidden query (the plain query is hidden, such that the server cannot "see" it) and query isolation (the server is allowed to "see" only the results - in encrypted format - in the search process). The scheme proposed in [1] is a SSE scheme. The first PEKS scheme was proposed in [2], in which the authors used billinear Diffie-Hellman or trapdoor permutation to generate the public key. the scheme in [2] is not too efficient, because the desired keyword is compared with every encrypted keyword in all documents.

In [3], the authors proposed the scheme PRISM, in which the task of searching for keywords is expressed as parallel instances of private information retrieval (PIR) for small amounts of data. The MapReduce model is used in this way: the mappers will solve the PIR instances, resulting a list of (key, value) pairs, while the reducers will take this list and aggregate it.

In [4], the authors proposed a modified version of MapReduce, namely M^2R, based on two remarks: 1) on an arbitrary node, the code that performs I/O or management work (such as scheduling) may be outside of TCB. Based on this remark, the authors developed four algorithms compatible with the MapReduce model, representing the trusted logic in TCB that will be ran by every node into a secure environment. 2) the data is exchanged and executed between mappers and reducers following a specific procedure. Based on this remark, the authors constructed a secure shuffle with a high level of security and more performing that ORAM solutions.

The recent works focuses on *forward and backward privacy* for dynamic SE schemes. Forward privacy means that the upload operation does not reveal *a priori* any information about what is contains (update is also performed on encrypted data) and backward privacy means that the delete operation does not reveal *a posteriori* any information about the deleted elements (delete is also performed on encrypted data). Some works that achieve forward and/or backward privacy are: [5], [6], [7].

3. The Proposed Framework

In this part of the study we describe how our framework works. Let's suppose that the data owner generated the pair of keys using *KeyGen* for an arbitrary encryption scheme, encrypted the documents that he/she owns using the *Enc* algorithm of the chosen encryption scheme and stored the documents in an encrypted format on a cloud server, which will organize the encrypted document as described in subsection *mapreduce* in order to be processed by mappers and reducers.

After these procedures, an authorized data user wants to retrieve from the server some documents that contain one ore more chosen keywords. To do this, the data user firstly computes the trapdoor for each keyword, then submits the trapdoors to the cloud server. The server will apply the *Test* algorithm embedded into MapReduce model. In the map phase, every trapdoor of keywords will receive the value 0 (because the search have not started). In the reduce model, every reducer will check if its *InputSplit* is compatible with (some of) the trapdoors, by running the *Test* algorithm of PEKS for each trapdoor. For every value of 1, the *ID* of the document is added to a list associated with the trapdoor.

Below, we present the algorithm.

Table 1. The implementation for Map function

```
Algorithm1. MapReduce model for PEKS

procedure prepare_input
// This procedure prepares the input for MapReduce model.
// It consist in running the algorithms in PEKS, except for Test.
end procedure

function map($enckw$):
input: a list of trapdoors of keywords $enckw$
output: a pair of form $(kw_i, 0)$
  for each element $kw_i$ in $enckw$:
    set $(kw_i, 0)$
  end for each
end function
```

Table 2. The implementation for Reduce function

```
Algorithm2. MapReduce model for PEKS

function reduce($kw, d$):
input: a trapdoor $t_{kw}$ and a document $d$
output: a pair of form $(kw, list)$
  for each encrypted word $kw_i$ in $d$:
    $list \leftarrow [\,]$
    if Test($k_p, t_{kw}$, Enc($k_p, kw_i$)) == 1
      then $list$.add($d$.ID)
    end if
  end for each
return $(t_{kw}, list)$

procedure send_output
// This procedure sends the output to user.
// The output represents all documents for which the list is not empty.
end procedure
```

After the data user receives the encrypted documents, he/she will decrypt them, applying the decryption algorithm and using the secret key.

4. Conclusions

Cloud computing is a main technology nowadays, but its security issues generate some concerns in adopting it. A response in this direction is searchable encryption. In this paper we proposed a general framework based on MapReduce model, which will speed up the search process of a searchable encryption scheme. This work is under development and practical result will be provided.

References

[1] Song, D. X., Wagner, D., & Perrig, A. Practical techniques for searches on encrypted data. In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000 (pp. 44-55). IEEE (2000)

[2] Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. Public key encryption with

keyword search. In International conference on the theory and applications of cryptographic techniques (pp. 506-522). Springer, Berlin, Heidelberg (2004)

[3] Blass, E. O., Di Pietro, R., Molva, R., & Önen, M. PRISM–privacy-preserving search in MapReduce. In International Symposium on Privacy Enhancing Technologies Symposium (pp. 180-200). Springer, Berlin, Heidelberg (2012)

[4] Dinh, T. T. A., Saxena, P., Chang, E. C., Ooi, B. C., & Zhang, C. (2015). M2R: Enabling stronger privacy in MapReduce computation. In 24th USENIX Security Symposium (USENIX Security 15) pp. 447--462 (2015)

[5] Zuo, C., Sun, S. F., Liu, J. K., Shao, J., & Pieprzyk, J. Dynamic searchable symmetric encryption schemes supporting range queries with forward (and backward) security. In European Symposium on Research in Computer Security (pp. 228-246). Springer, Cham (2018)

[6] Bakas, A., & Michalas, A. Multi-Client Symmetric Searchable Encryption with Forward Privacy. Cryptology ePrint Archive (2019)

[7] Zuo, C., Sun, S. F., Liu, J. K., Shao, J., & Pieprzyk, J. Dynamic Searchable Symmetric Encryption with Forward and Stronger Backward Privacy. In European Symposium on Research in Computer Security (pp. 283-303). Springer, Cham (2019)